Lab 4. Tableau Desktop Advanced Calculations

In this lab, we will cover the following topics:

- Creating calculated fields
- Implementing quick table calculations
- Creating and using groups
- Creating and using sets
- Creating and using parameters
- Implementing the basics of level of detail expressions
- Using custom geocoding
- Using polygons for analytics

Creating calculated fields

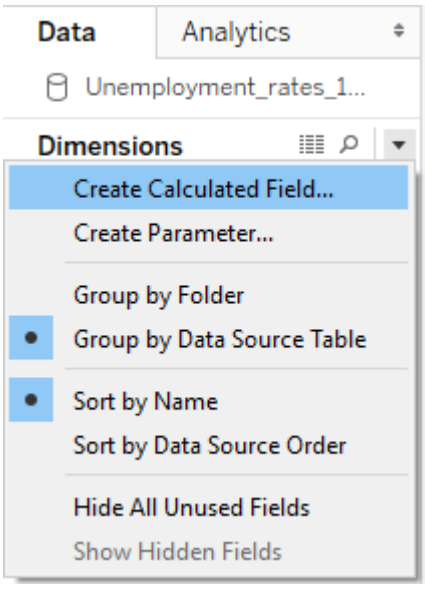
Calculated fields are custom fields you can add to your data source. Their values are determined by the custom formula you yourself write. In this recipe, we will create a simple calculated field that contains the difference in value of **C``ivilian`` Labor Force** between each data point and the data point preceding it.

Getting ready

Connect to the `Unemployment_rates_1990-2016.csv` dataset, and open a blank worksheet.

How to do it...

1. Click on the black drop-down arrow in the **Data** pane, to the right of **Dimensions**.
2. Select **Create Calculated Field...**. Alternatively, from the main menu toolbar select **Analysis**, and choose **Create Calculated Field**:

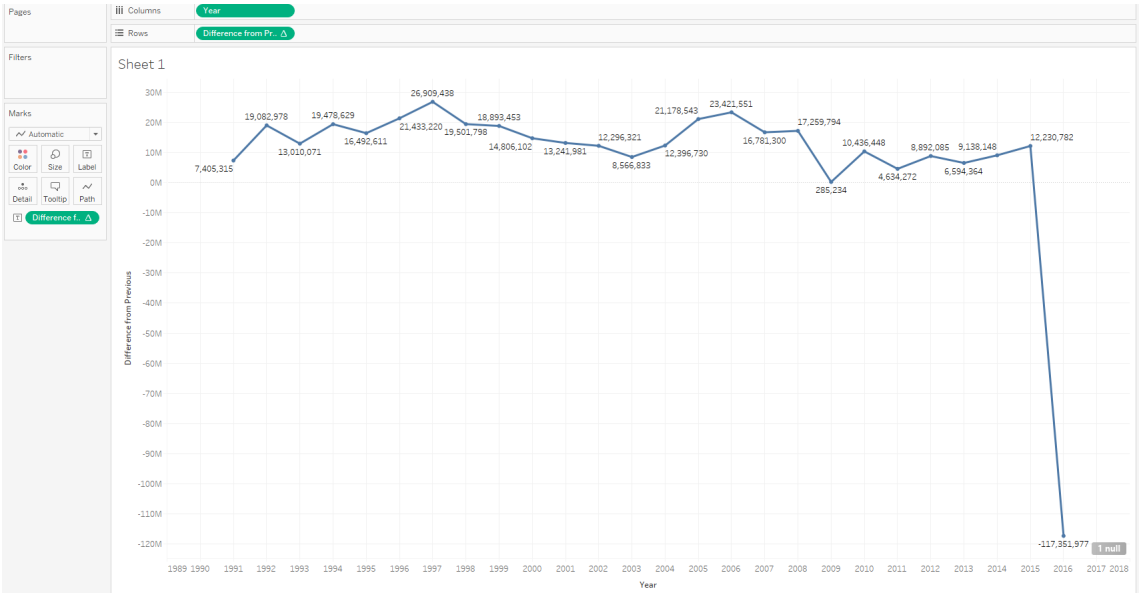


3. Rename the field from **Calculation 1** to `Difference from Previous`. Click on the **Apply** button and then click on **OK**.
4. In the formula pane, type the following expression and click on the **Apply** button and then click on **OK**:

```
SUM([Civilian Labor Force])-LOOKUP(SUM([Civilian Labor Force]),-1)
```

Let's see how it looks in the following screenshot:

5. The new calculated field, **`Difference from Previous`**, has appeared under **`Measures`**.
6. Drag and drop the new calculated field into **`Text`** in the **`Marks`** cards.
7. Drag and drop the new calculated field into the **`Rows`** shelf.
8. Drag and drop **`Year`** **from** **`Dimensions`** into the **`Columns`** shelf:



How it works...

We created a calculated field that contains the difference between the current and the previous data point. We used the `LOOKUP` function to return the aggregated value of `Civilian Labor Force` for the previous data point, and then subtracted it from the aggregated value of `Civilian Labor Force` in the current data point. This resulted in a new field that contains this difference, and which we can now use in our visualizations. Keep in mind that all table calculation functions, including the `LOOKUP` function, require an aggregated value to work with. That

means that, when we use fields without an aggregation function -- for example, just `Civilian Labor Force` instead of `SUM([Civilian Labor Force])` ---we may get an error in the calculated field.

However, it is important to note that our original dataset has remained unchanged---the calculated field we created has not been written to it and, if we were to open a new Tableau workbook and connect to the `Unemployment_rates_1990-2016.csv` dataset, we would not find our calculated field among the fields in the data source.

There's more...

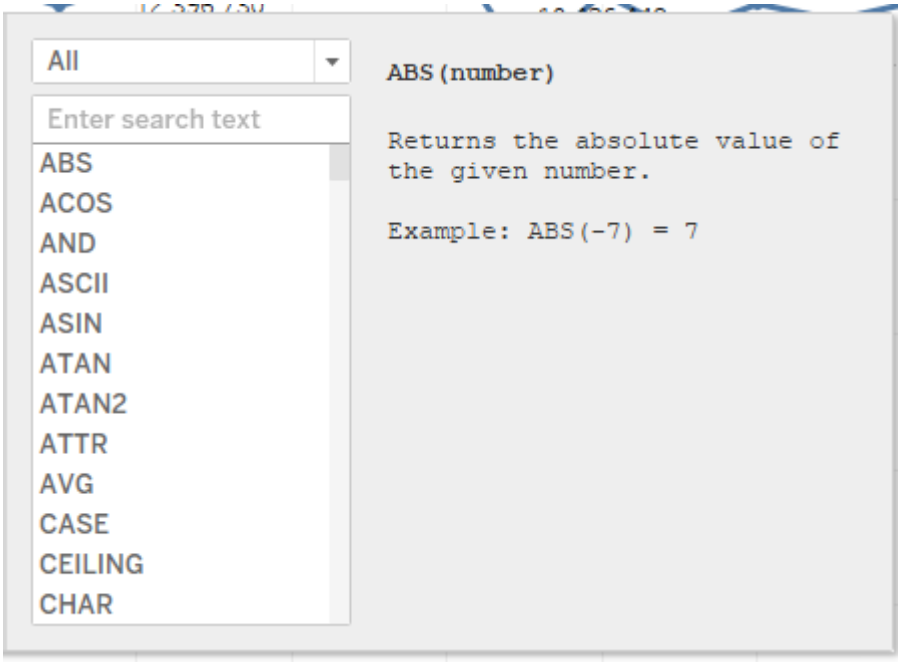
The type of calculation we created in this recipe is called table calculation, which simply means that the value of the calculated field we created is dependent on the dimension we use with it in our visualization. If we were, for example, to choose **`Month`** instead of **`Year`**, we would get the difference from the previous month. If we look at the **`Data Source`** sheet, we will see that the value of this calculated field in each row is **`Undefined`**, as shown in the following screenshot:

Area	Year	Month	Civilian Labor Force	Employment	Unemployment	Unemployment Ra...	Difference from pr...
Anniston-Oxford-Jack...	1990	1	51,485	48,307	3,178	6.2000	*Undefined*
Auburn-Opelika, AL M...	1990	1	44,415	41,247	3,168	7.1000	*Undefined*
Birmingham-Hoover, ...	1990	1	457,612	433,590	24,022	5.2000	*Undefined*
Daphne-Fairhope-Fole...	1990	1	45,859	43,402	2,457	5.4000	*Undefined*
Decatur, AL MSA	1990	1	65,452	61,009	4,443	6.8000	*Undefined*
Dothan, AL MSA	1990	1	58,423	56,097	2,326	4.0000	*Undefined*
Florence-Muscle Shoa...	1990	1	61,752	57,486	4,266	6.9000	*Undefined*

However, calculated fields can also have other types of output. We could, for example, sum up **`Employment`** and **`Unemployment`** to get **`Civilian Labor Force`**, using the `[Employment]+[Unemployment]` expression. If we look at the **`Data Source`** preview now, we can see the value of the new calculated field for each row in our data source:

Area	Year	Month	Civilian Labor Force	Employment	Unemployment	Unemployment Ra...	Difference from pr...	Calculated Labor Force
Anniston-Oxford-Jack...	1990	1	51,485	48,307	3,178	6.2000	*Undefined*	51,485
Auburn-Opelika, AL M...	1990	1	44,415	41,247	3,168	7.1000	*Undefined*	44,415
Birmingham-Hoover, ...	1990	1	457,612	433,590	24,022	5.2000	*Undefined*	457,612
Daphne-Fairhope-Fole...	1990	1	45,859	43,402	2,457	5.4000	*Undefined*	45,859
Decatur, AL MSA	1990	1	65,452	61,009	4,443	6.8000	*Undefined*	65,452
Dothan, AL MSA	1990	1	58,423	56,097	2,326	4.0000	*Undefined*	58,423

Tableau offers a multitude of functions that can be used to perform a very versatile and wide range of tasks. To truly master Tableau calculated fields, one must understand and get comfortable with using functions. Every function in Tableau has certain arguments it requires. To view functions, their descriptions, and the arguments they take, open a new calculated field and click on the arrow on the right-hand side of the editor window. This will expand the window to display a list of functions with their associated details:



You should also always be mindful of the fact that not all functions can be applied to all data types. For example, the `SUM` function cannot be applied to strings, but can be applied to numerical values. In the drop-down menu on top of the list of functions, you can select functions suited for different data types.

Implementing quick table calculations

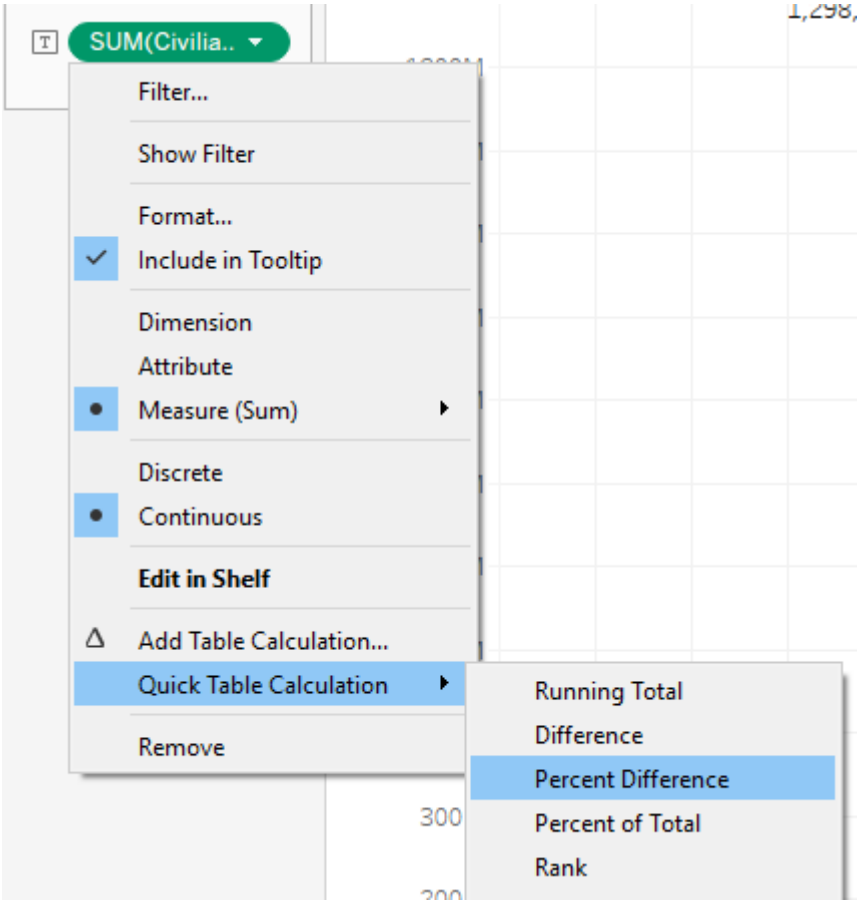
Table calculations are a type of calculated fields. They allow you to transform the values in your view, taking into account only the values that are currently in the view (and not considering the ones that are filtered out). In the previous recipe, we have created a table calculation through a calculated field. Quick table calculations are a handy way to implement common table calculations into your view, without setting them up manually.

Getting ready

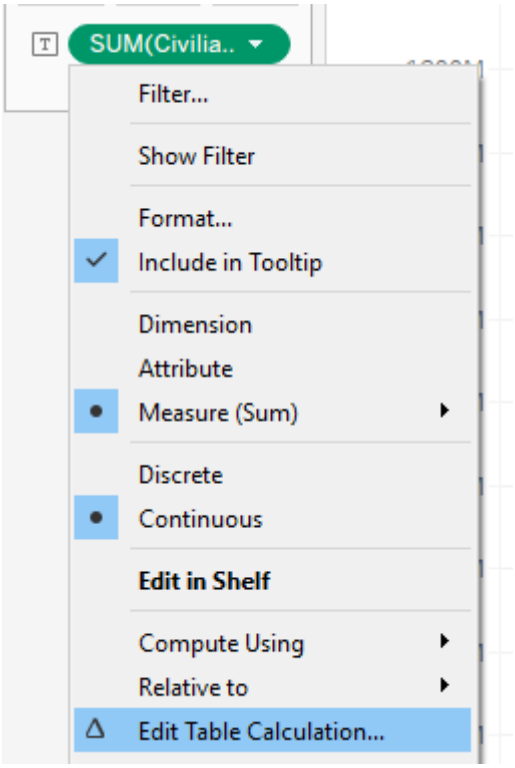
In order to follow this recipe, you will need to connect to the `Unemployment_rates_1990-2016.csv` dataset, and open a blank worksheet.

How to do it...

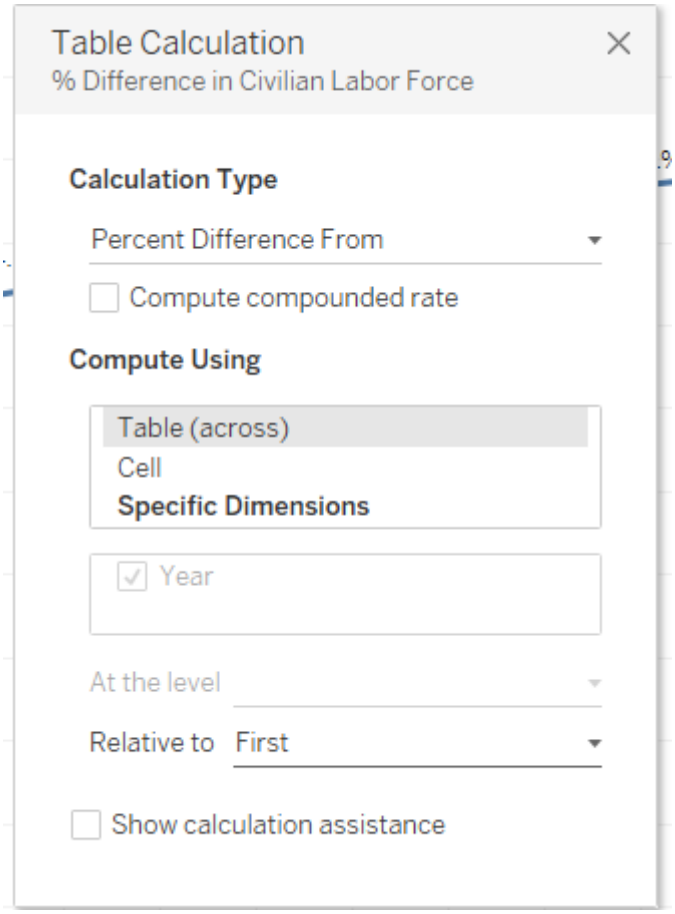
1. Drag and drop `Year` from `Dimensions` into the `Columns` shelf.
2. Drag and drop `Civilian Labor Force` from `Measures` into the `Rows` shelf.
3. Again, drag and drop `Civilian Labor Force` from `Measures` onto `Label` in the `Marks` card.
4. Hover over the `Civilian Labor Force` pill in `Text` and click on the white arrow that appears on it.
5. In the drop-down menu, navigate to `Quick Table Calculation` | `Percent Difference` as shown in the following screenshot:



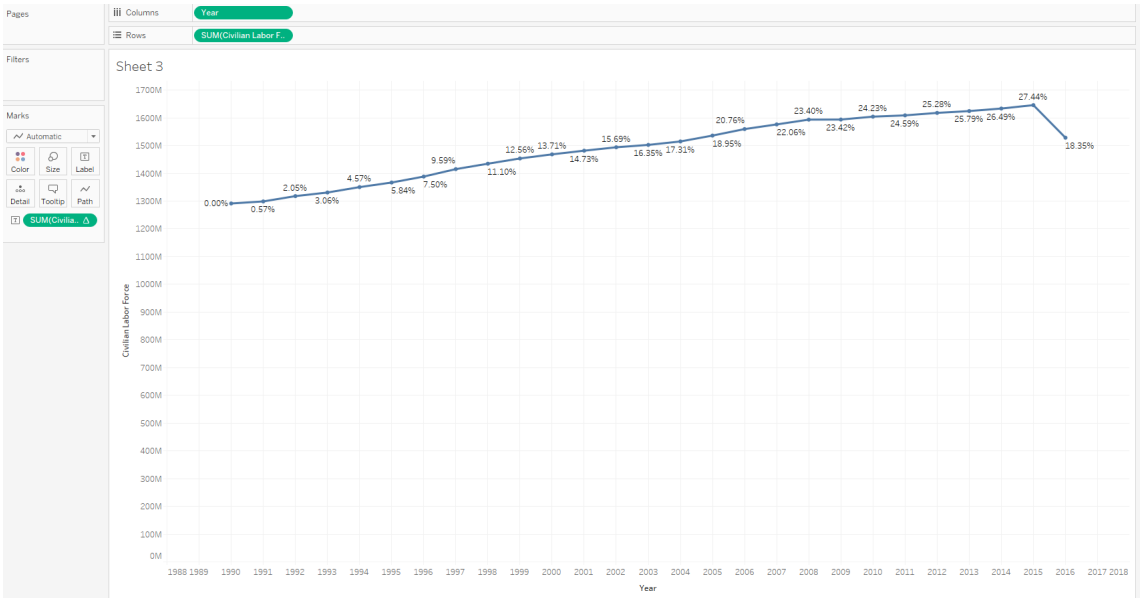
6. Hover over the `Civilian Labor Force` pill again, and click on the white arrow that appears on it.
7. From the drop-down menu, select `Edit Table Calculation...` as shown in the following screenshot:



8. In the `Table Calculation` window that appears, click on the `Relative to` drop-down menu in the bottom, and select `First`:



9. When you are done, close the window:



How it works...

Quick table calculations allow for quick implementation of common transformations, with the default setting. In this case, we implemented the **`Percent Difference`** calculation. By default, it sets the difference to be relative to the previous data point (**`Year`**). Then, we edited the default settings a bit by changing the calculation to be relative to the first data point in the view, not the previous one. Because of that, all of the percentages in our view are set to be relative to the year 1990. However, it is important to note that table calculations only take into account the data that is in the view. So, for example, if we filtered the year 1990 out from our view, all our calculations would then be relative to the year 1991, which would become the first data point in the view.

There's more...

Quick table calculations are just shortcuts to implementing common table calculations. Table calculations can also be implemented manually. To implement a table calculation from scratch, click on the pill of the measure you would like to transform (in this recipe, **`Civilian Labor Force`**), and select **`Add Table Calculation`** from the drop-down menu. In the **`Table Calculation`** window that opens, you can adjust all the relevant parameters based on the calculation type you select.

Creating and using groups

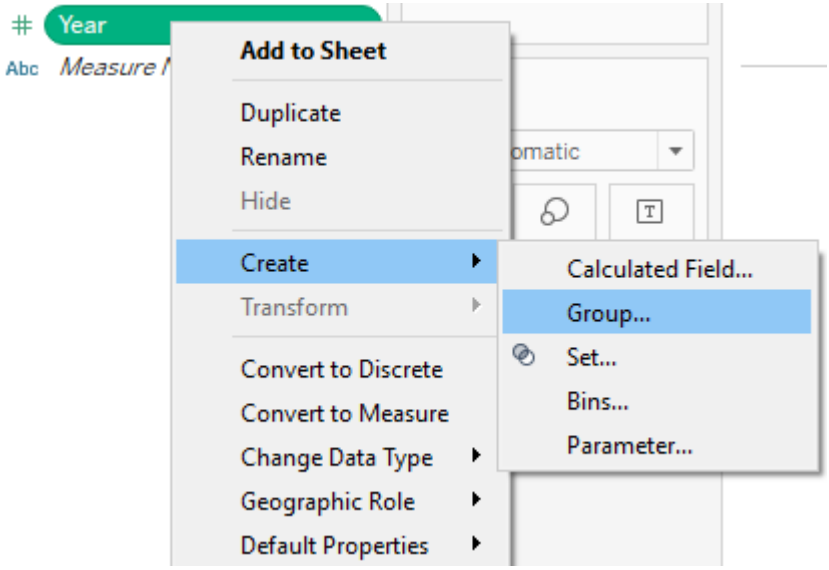
Groups are a handy way to combine multiple members of a field into a new member. In the example we will be working on in this recipe, we will group individual years into decades, so that we are able to see the data on the level of decade.

Getting ready

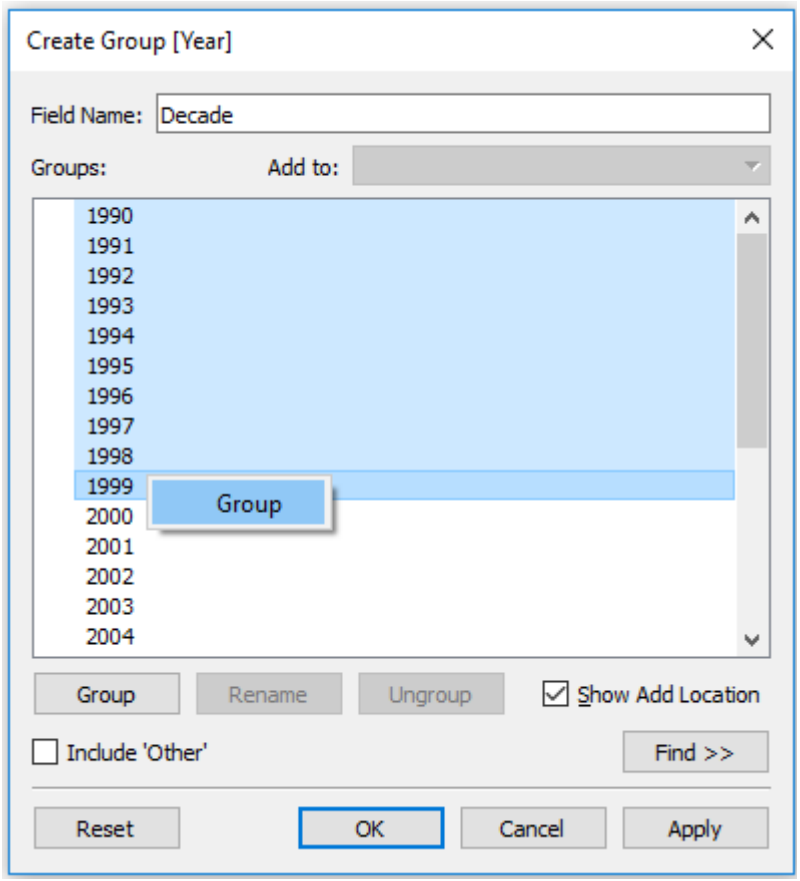
Connect to your local copy of the `Unemployment_rates_1990-2016.csv` dataset, and open a new blank worksheet.

How to do it...

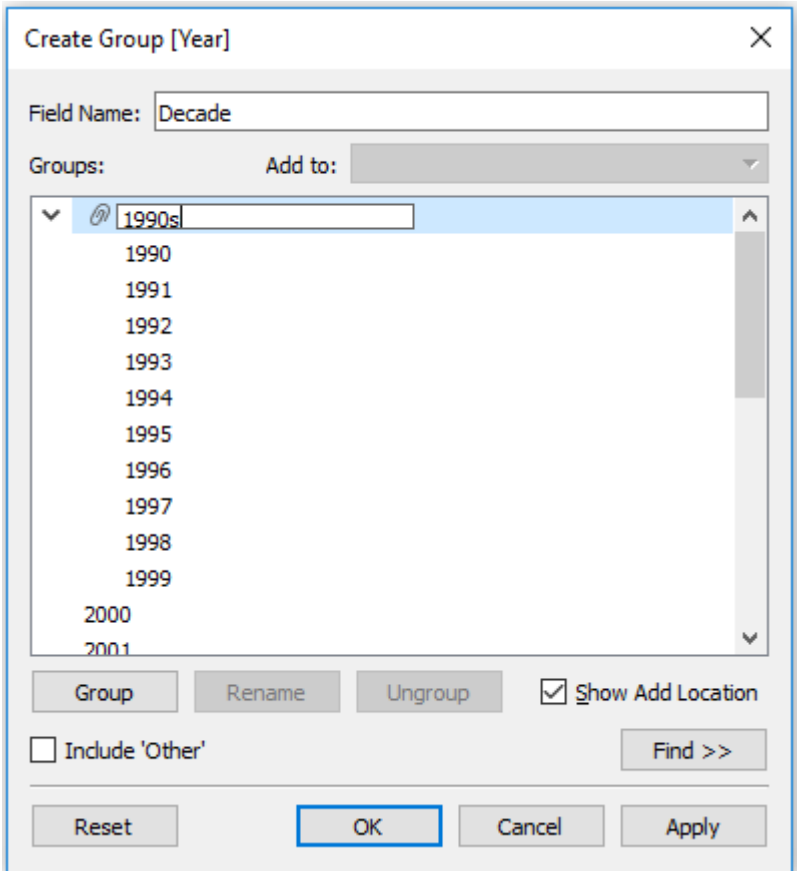
1. Right-click on the **`Year`** field under **`Dimensions`** .
2. In the drop-down menu, navigate to **`Create`** | **`Group`**... :



3. In the **`Field Name`** field, change the name of the group to `Decade`.
4. In the **`Create Group [Year]`** window, select years **`1990`** through **`1999`** by clicking on the first and the last year while holding the [*Shift*] button on the keyboard.
5. When the years are selected, click on **`Group`** at the bottom of the list or, alternatively, right-click on the selection and select **`Group`**:



6. A new group will appear, called **`1990, 1991, 1992 and 7 more`**. The name of the group will be enabled for editing by default. Rename the group to `1990s` as shown in the following screenshot:

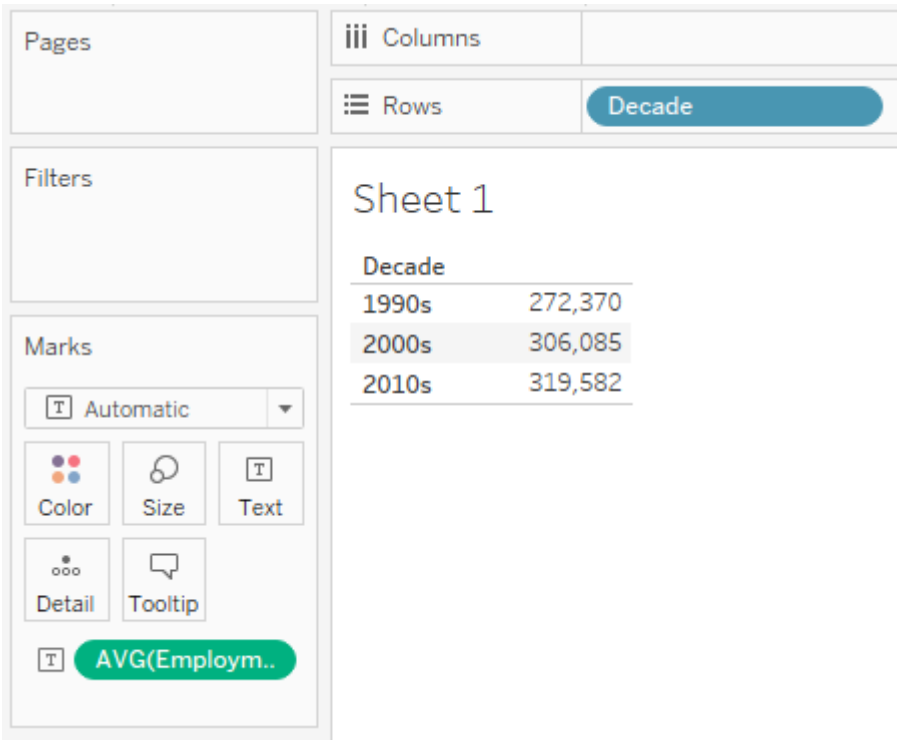


7. Now, select years `2000` through `2009` and click the **`Group`** button.
8. Rename the group to `2000s`.
9. Finally, select the remaining years--- **`2010`** through **`2019`** and click on the **`Group`** button to group them.
10. Rename the group to `2010s`.
11. When all three groups are created and renamed, click **`OK`**.
12. The new group has now appeared under **`Dimensions`** in the **`Data`** pane, denoted by a paperclip symbol:

    ```
    . Drag and drop **`Decade`** into the **`Rows`** shelf.
    ```

13. Drag and drop **`Employment`** from **`Measures`** onto **`Text`** in the **`Marks`** card.
14. Hover over the **`AVG(Employment)`** pill in the ** `Marks` **card and click on the white arrow that appears on it.
15. Navigate to **`Measure (Sum)`** and select **`Average`**:

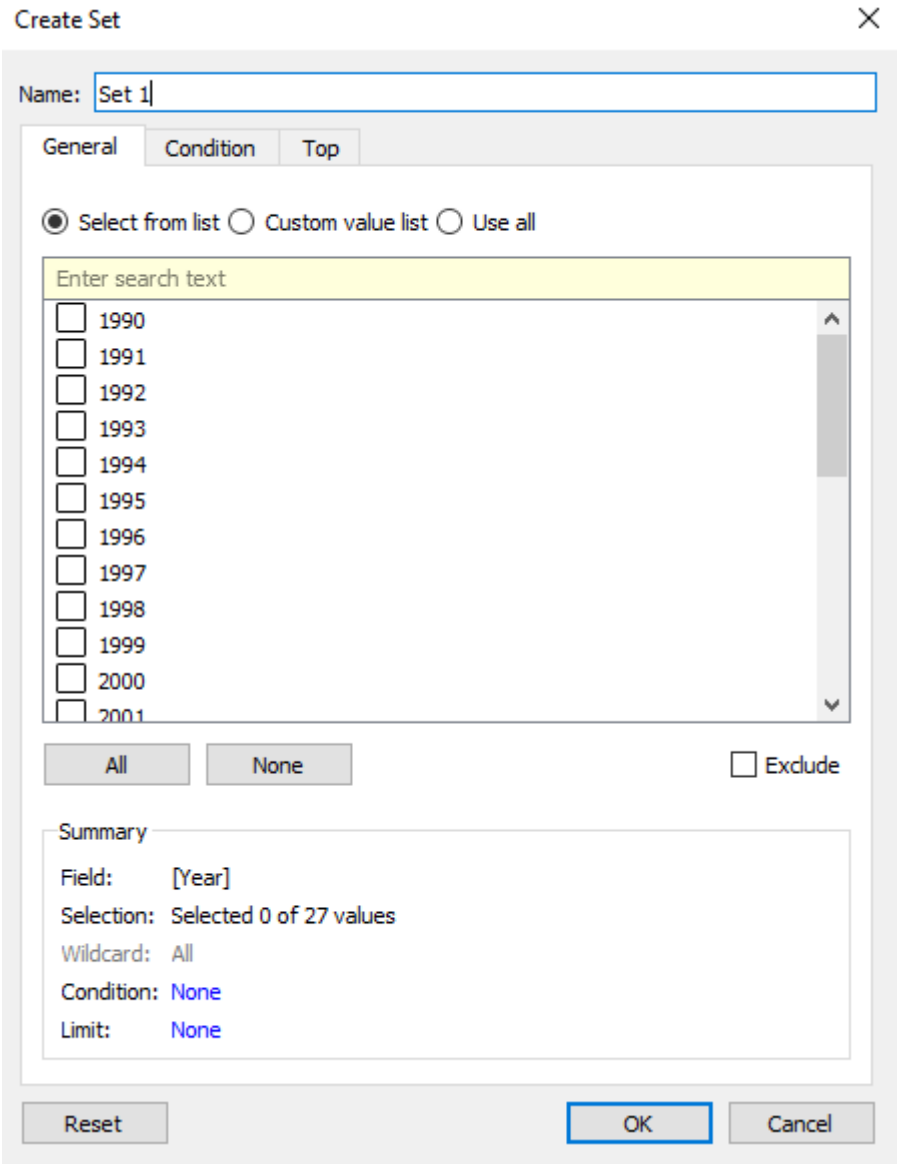


How it works...

Groups allow us to analyze our data on an aggregated level that is not available in the raw dataset we are using. By grouping, we have created a new group and field. When we insert it into a view and a measure (in this case, `Employment`), the measure is aggregated on the level of group. This means that, if we use the `Average` function, Tableau returns a single average value calculated across all the rows that belong group member (in our case, `Decade`).

There's more...

When we do not want to classify all the field members into groups, Tableau allows us to group all remaining members into a single, `Other`, category. When you are done creating the groups you want to have in your new group field, tick the box in front of `Include 'Other'` in the bottom of the `Create Group` window. Tableau will group all the ungrouped members into a single group, named `Other`:



Creating and using sets

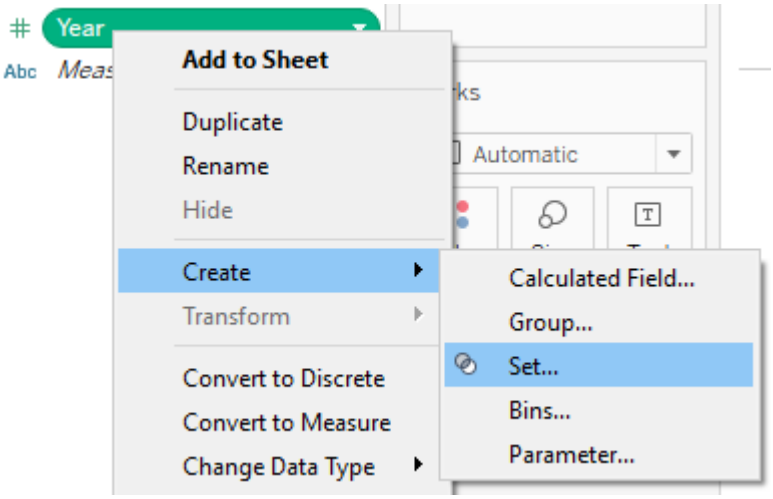
Sets are, in a sense, the opposite of groups. We use them to slice our data: subset our data based on a condition, and explore it on a more granular level. In the following recipe, we will create a set that makes a distinction between years that had above average employment, and years that had below average employment.

Getting ready

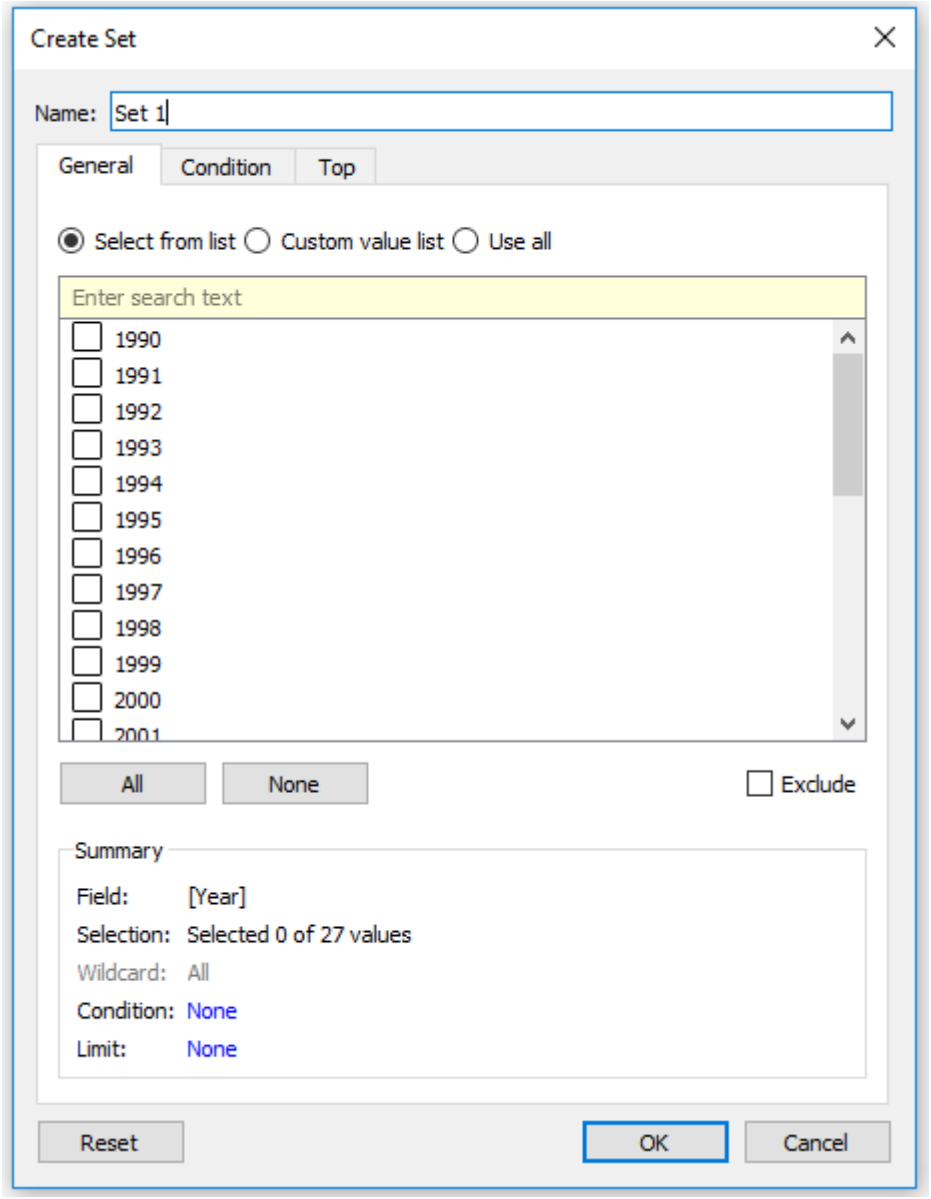
Connect to the `Unemployment_rates_1990-2016.csv` dataset, and open a blank worksheet.

How to do it...

1. Right-click on the **`Year`** field under **`Dimensions`**.
2. In the drop-down menu, navigate to **`Create`** | **`Set...`** as shown in the following screenshot:



3. In the **`Name`** field, change the name of the set from **`Set 1`** to `Above average employment years`:



4. Click on the **`Condition`** tab and select **`By field`**.

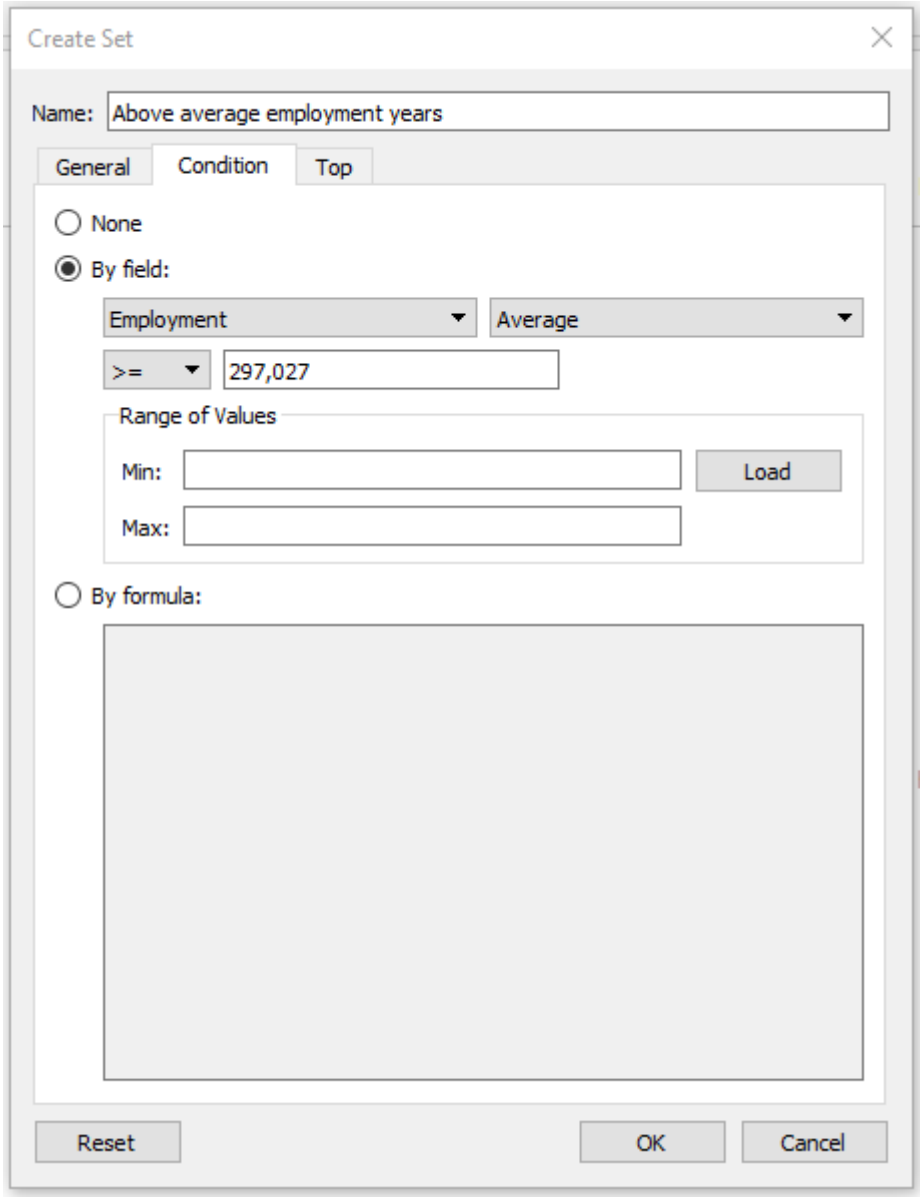
5. From the first drop-down menu, choose **`Employment`**.

6. From the second drop-down menu, select **`Average`**.

7. From the third drop-down menu, select the **`>=`** (greater or equal to) sign.

8. In the number field, type `297027`. This is the overall average of employment for all years.

9. Click **`OK`** to exit the window:



10. A new section, **`Sets`**, has appeared in the **`Data`** pane, below **`Measures`**. It contains our new set, denoted by the set symbol

    ```
    . Drag and drop the set we created onto **`Color`** in the
    ```

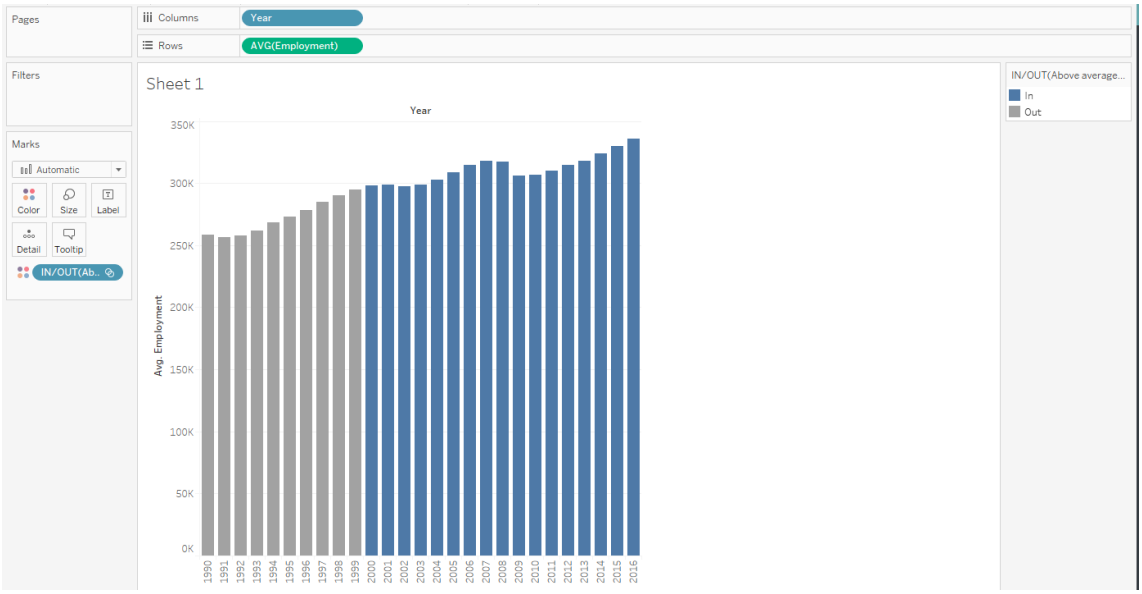
    `Marks` label.

11. Drag and drop **`Year`** from **`Dimensions`** into the **`Columns`** shelf.

12. Right-click on the **`Year`** pill in the **`Columns`** shelf, and in the drop-down menu switch to **`Discrete`**.

13. Drag and drop **`Employment`** **from** **`Measures`** into the **`Rows`** shelf.

14. Hover over the **`Employment`** pill in the ** `Rows` **shelf and click on the white arrow that appears on it.
15. In the drop-down menu, navigate to **`Measure (Sum)`** and select **`Average`**. This will provide the following output:



How it works...

In this recipe, we have created a set based on a condition: the value of **`Employment`**. We have set the condition so that all the data points that have the value of **`Employment`** equal to or higher than **`297027`** (which is the average value of **`Employment`** across the entire dataset) are included in the set, and all the other data points are excluded from it. This created a new field---our set, which we then used in the visualization. The set field now works as a discrete dimension with two values: **`In`** and **`Out`** that denote the data points that are in/out of the set. We used these values in our visualization to color the year with average or above average employment.

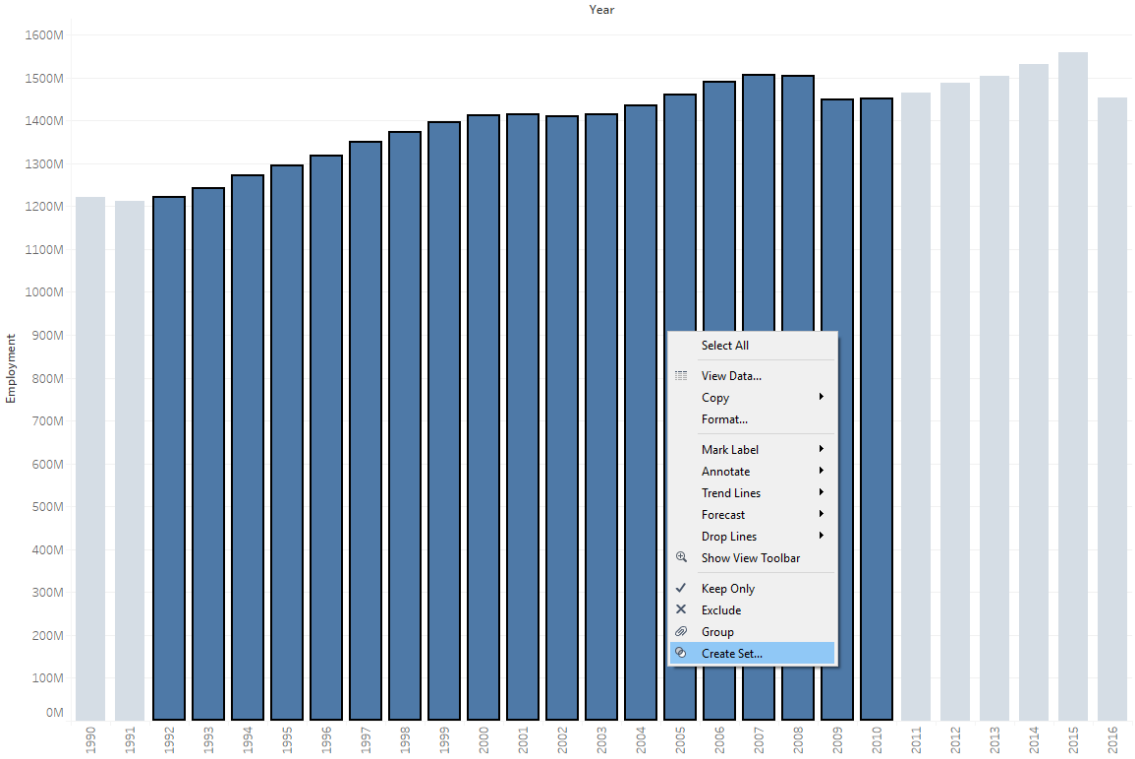
Note

If you want the set members to appear in your visualization, right-click on the **`Set`** pill that you have included in your view, and in the drop-down menu switch from **`S``how In``/``Out of Set`** to **`Show Members in Set`**.

There's more...

Tableau offers two types of sets: fixed and dynamic. The set we made in this recipe is a dynamic set. It means the members of the set will change with the underlying data. However, there is a constraint related to dynamic sets: they can only be based on one dimension.

On the other hand, fixed sets do not change: their members always remain the same. We can create a fixed set by simply selecting data point as our view, right-clicking on the selection, and choosing the ** `Create Set...` **option from the drop-down menu, as shown in the following screenshot:



Creating and using parameters

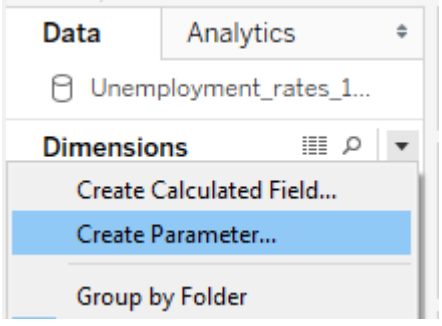
In the previous recipe, [*Creating and using sets*],[** **]we created a set to divide years into the ones with above average and the ones with below average employment. We did this by hardcoding the average value in the definition of our set. But, what if we wanted to make visualizations where we can input different values ourselves, so our visualization changes dynamically when we change the value? We can easily achieve this using parameters. Let's create a set where we can dynamically change the value that defines the set.

Getting ready

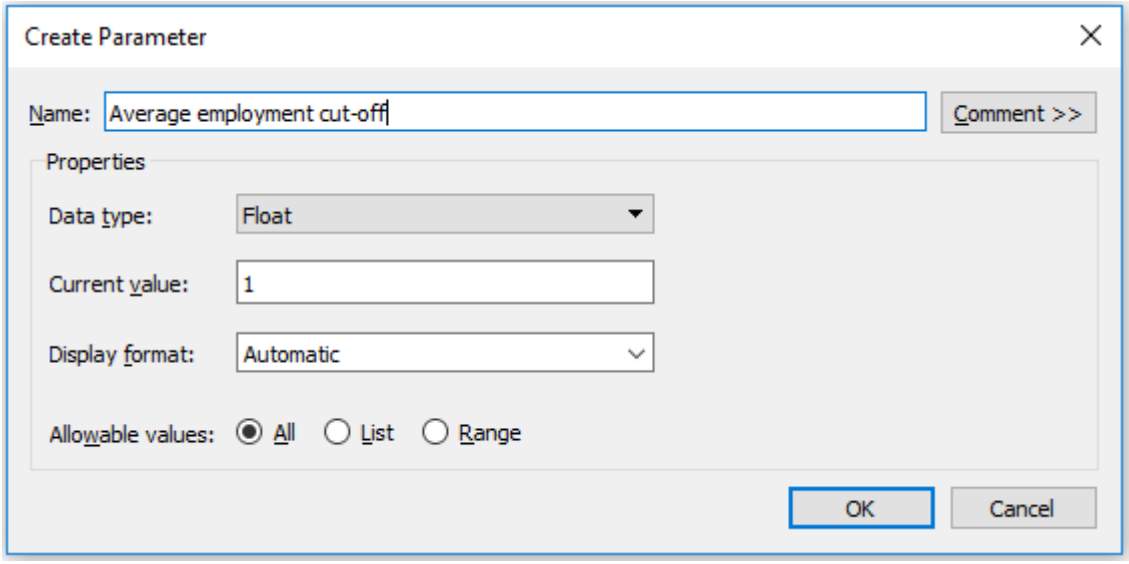
We will use the `Unemployment_rates_1990-2016.csv` dataset, so make sure you are connected to it, and open a new blank worksheet.

How to do it...

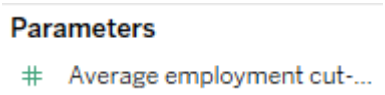
1. In the **`Data`** pane, click on the black arrow to the right of the **`Dimensions`** tab and select **`Create Parameter`**`...`:



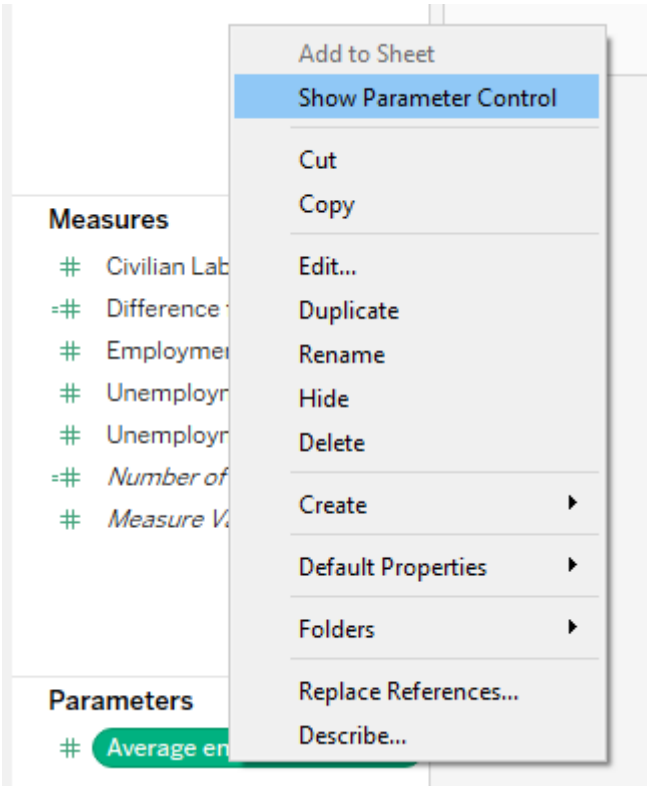
2. In the `Create Parameter` window, change the name of the parameter from `Parameter 1` to `Average employment cut-off`:



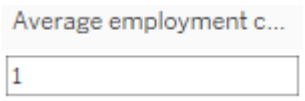
3. Leave all the other settings as they are, and click `OK` to exit the window. A new section, `Parameters`, will appear in the `Data` pane, beneath `Measures`. It will contain our new parameter, `Average employment cut-off`:



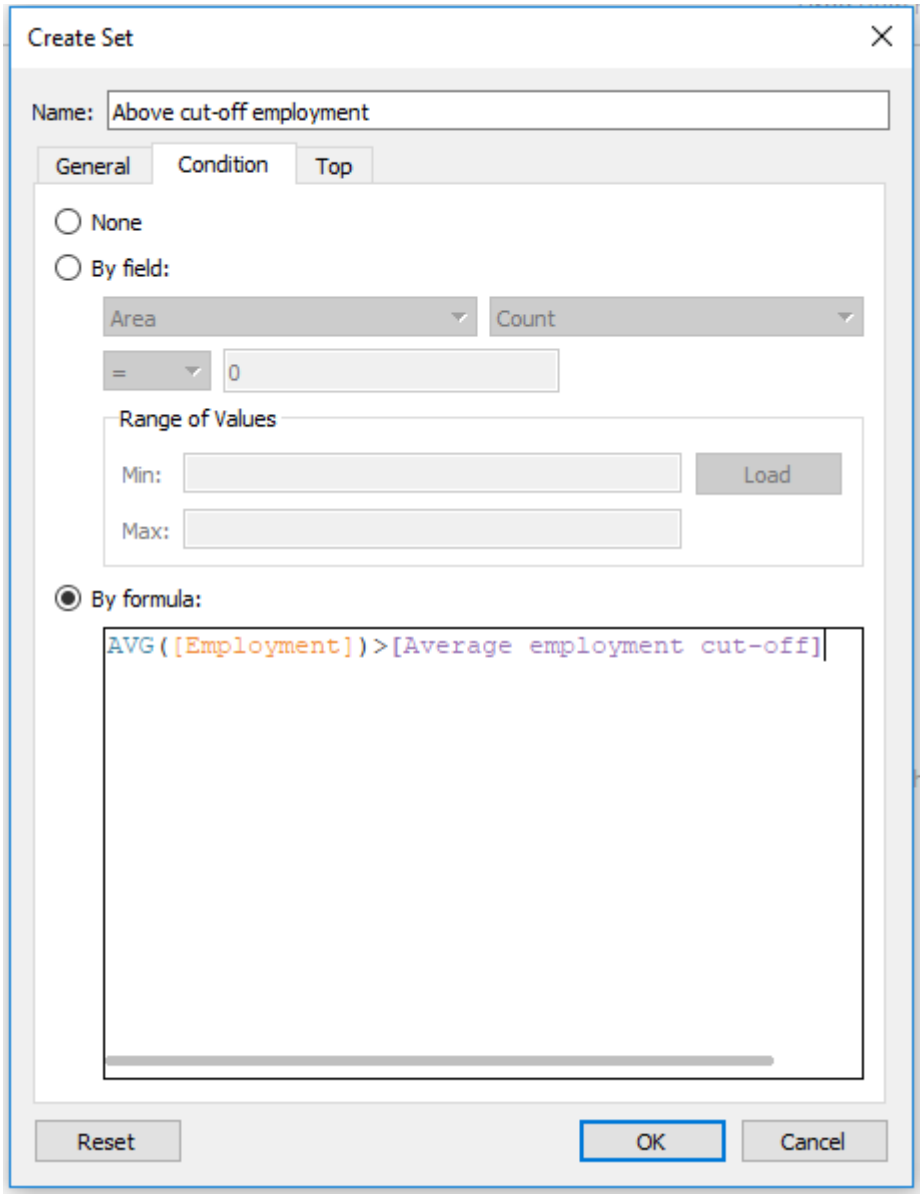
4. Right-click on the `Average Employment cut-off` parameter under `Parameters` and select `Show Parameter Control`:



The parameter control card will appear in the top-right corner of the workspace. Notice that it shows the value we have set as the current value when creating it, which is `1`:



5. We have created the parameter. Now, let's create the set we will use it with. Right-click on the **`Year`** pill under **`Dimensions`**, and navigate to **`Create`** | **`Set...`**.
6. In the **`Create Set`** window, change the name of the set from **`Set 1`** to `Above cut-off employment`.
7. Switch from the **`General`** tab to the **`Condition`** tab, and select **`By formula`**.
8. In the formula space, type the `AVG([Employment])>[Average employment cut-off]` expression as shown in the following screenshot:

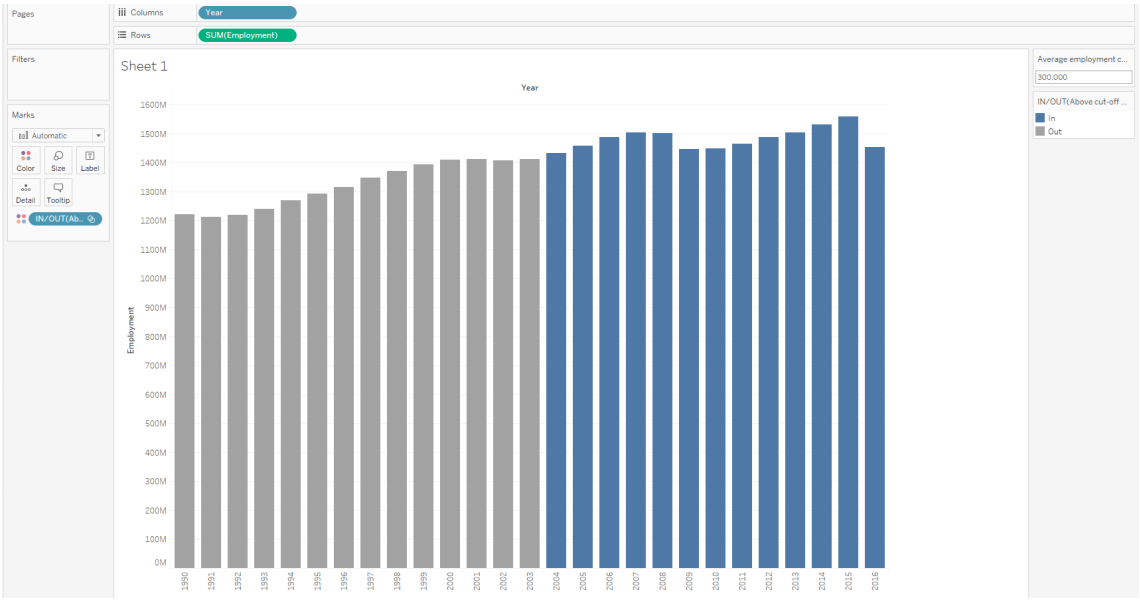


9. When you are done, click **`OK`** to close the window. Our set, **`Above cut-off employment`**, has now appeared together with a new **`Set`** section in the **`Data`** pane.
10. Finally, let's make a visualization using the new set. Drag and drop **`Year`** from **`Dimensions`** into the **`Columns`** shelf.
11. Right-click on the **`Year`** pill in the **`Columns`** shelf, and in the drop-down menu switch to **`Discrete`**.
12. Drag and drop ** `Employment` **from** `Measures` **into the** `Rows` **shelf.

Note

Although we have used average employment as the aggregation function when creating our set, we can leave the **`SUM`** aggregation function in our view. The two are unrelated, since the function in the set is used to define the set, and is not related to the measures we might use in our view.

13. Drag and drop the **`Above cut-off employment`** set from **`Sets`** onto **`Color`** in the **`Marks`** card.
14. All our bars are colored as **`In`** set, because the **`Average employment cut-off`** parameter's value is set to `1`. In the parameter control card, try changing the value to `300000`.
15. Try changing the value a couple of more times, to see how the years that are in/out of the set change with the parameter value:



How it works...

Parameters are values that can change dynamically based on user input. They can replace constant values, in a range of different cases.

In the preceding example, we created a parameter that takes a number as an input from the user. We then used the value of the user input as the threshold score for membership in the set, by setting a condition. So, instead of the constant (average) value we inserted as the threshold value for set membership in the previous recipe, now we have a dynamic value, which changes our set every time a user inputs a new number. So, every time we change the user input, different years are included in or excluded from the set.

There's more...

There are many ways in which we can use parameters to make our visualizations interactive. For example, we can use parameters to switch between different fields (measures or dimensions in our view) or create dynamic reference lines. Generally, when using parameters, you will mostly be using them together with calculated fields. Parameters themselves cannot be included in the view. They are included by creating a calculated field that takes advantage of the parameter, and then including the calculated field in the view.

Implementing the basics of level of detail expressions

[**Level of detail**] ([**LOD**]) expressions are a very useful feature of Tableau. In order to understand what LOD expressions do, we must first understand what LOD is. When we use one dimension in our view, Tableau aggregates the measures along this dimension; so, we say that this dimension provides the LOD. If we add another dimension to the view, measures will be additionally disaggregated along this dimension as well, leading to an even more granular

LOD. LOD expressions allow us to control the LOD that will be used in our view independently of the fields that are included in the view itself. We can make it either more or less granular than the LOD provided by the fields included in the visualization itself, which provides for flexibility in creating our views.

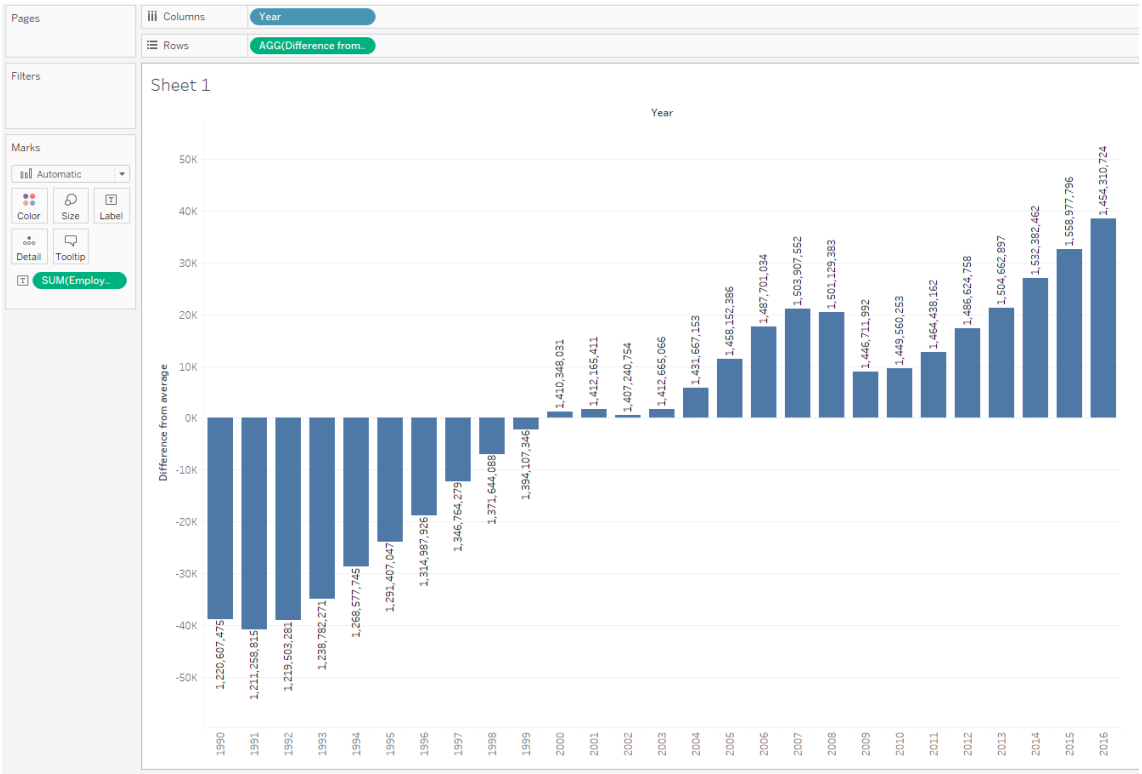
Getting ready

To follow the steps in this recipe, connect to the `Unemployment_rates_1990-2016.csv` dataset, and open a new blank worksheet.

How to do it...

1. In the **Data** pane, click on the black arrow to the right of **Dimensions** and select **Create Calculated Field...** .
2. Rename the new calculated field from **Calculation 1** to `Difference from average` .
3. In the formula space, type `AVG([Employment])-{FIXED: AVG([Employment])})` , as shown in the following screenshot:

4. Click on **Apply** and then click on **OK** to exit the window. The new calculated field, **Difference from average** , has now appeared under **Measures** in the **Data** pane.
5. Drag and drop **Difference from average** from **Measures** into the **Rows** shelf.
6. Drag and drop **Year** from **Dimensions** into the **Columns** shelf.
7. Right-click on the **Year** pill in the **Columns** shelf, and in the drop-down menu switch to **Discrete** .
8. Drag and drop **Employment** from **Measures** onto **Label** in the **Marks** card as shown in the following screenshot:



How it works...

In this recipe, we created a visualization where difference from the overall average of employment is calculated, but in such a way that the overall average is used as a reference point regardless of what is included in the visualization. So, when we include **`Year`**, our visualization shows the average difference in **`Employment`** per year (when we say average, we mean average across all the rows in the data source that belong to the particular year) from the overall average of **`Employment`**.

If we, for example, included **`Month`** in our visualization as well, the average difference would be computed for each month of each year, but would still be calculated relative to the overall average of **`Employment`**. That's because we told Tableau to keep the overall average fixed, regardless of what is in our visualization.

There's more...

In this recipe, we used a `FIXED LOD` expression. With the `FIXED LOD` expression, we specify the exact LOD we want to have in our visualization, and it is completely unaffected by the LOD we include in our view. Because of this, we kept the overall average of **`Employment`** in our calculation, despite introducing other dimensions into the view that would normally disaggregate our measure.

Tableau also offers the following two other types of LOD expressions. Let's take a quick look at what they do:

- `INCLUDE`: These expressions allow us to introduce an additional LOD into the view that is more granular than the one determined by the visualization itself. The `INCLUDE` expressions take into account the LOD that is included in the visualization, but always disaggregates data additionally by the dimension we specified in the `INCLUDE` expression itself.
- `EXCLUDE`: These expressions do the opposite of the `INCLUDE` expressions---they keep the data at the certain level of granularity that is higher than the one introduced by the view itself. This means that, when we create our visualization, Tableau will aggregate the measure according to the view, with the exception of

the dimension that is specified in the `EXCLUDE` expression. This will result in the values repeating across multiple data points in our view because, instead of aggregating them across these data points, they are aggregated at a higher level.

Implementing the basics of level of detail expressions

[**Level of detail**] ([**LOD**]) expressions are a very useful feature of Tableau. In order to understand what LOD expressions do, we must first understand what LOD is. When we use one dimension in our view, Tableau aggregates the measures along this dimension; so, we say that this dimension provides the LOD. If we add another dimension to the view, measures will be additionally disaggregated along this dimension as well, leading to an even more granular LOD. LOD expressions allow us to control the LOD that will be used in our view independently of the fields that are included in the view itself. We can make it either more or less granular than the LOD provided by the fields included in the visualization itself, which provides for flexibility in creating our views.

Getting ready

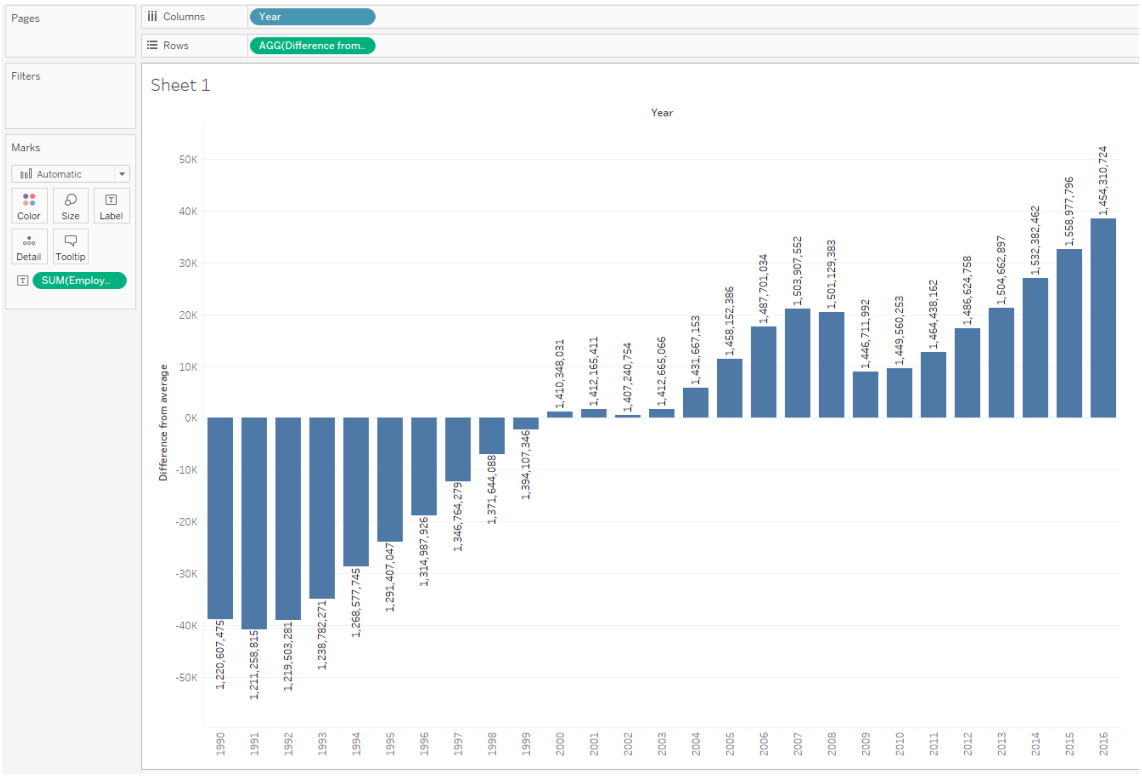
To follow the steps in this recipe, connect to the `Unemployment_rates_1990-2016.csv` dataset, and open a new blank worksheet.

How to do it...

1. In the **`Data`** pane, click on the black arrow to the right of **`Dimensions`** and select **`Create Calculated Field...`**.
2. Rename the new calculated field from **`Calculation 1`** to `Difference from average`.
3. In the formula space, type `AVG([Employment])-{FIXED: AVG([Employment])})`, as shown in the following screenshot:

4. Click on **`Apply`** and then click on **`OK`** to exit the window. The new calculated field, **`Difference from average`**, has now appeared under **`Measures`** in the **`Data`** pane.
5. Drag and drop **`Difference from average`** from **`Measures`** into the **`Rows`** shelf.
6. Drag and drop **`Year`** from **`Dimensions`** into the **`Columns`** shelf.
7. Right-click on the **`Year`** pill in the **`Columns`** shelf, and in the drop-down menu switch to **`Discrete`**.

8. Drag and drop **`Employment`** from **`Measures`** onto **`Label`** in the **`Marks`** card as shown in the following screenshot:



How it works...

In this recipe, we created a visualization where difference from the overall average of employment is calculated, but in such a way that the overall average is used as a reference point regardless of what is included in the visualization. So, when we include **`Year`**, our visualization shows the average difference in **`Employment`** per year (when we say average, we mean average across all the rows in the data source that belong to the particular year) from the overall average of **`Employment`**.

If we, for example, included **`Month`** in our visualization as well, the average difference would be computed for each month of each year, but would still be calculated relative to the overall average of **`Employment`**. That's because we told Tableau to keep the overall average fixed, regardless of what is in our visualization.

There's more...

In this recipe, we used a `FIXED LOD` expression. With the `FIXED LOD` expression, we specify the exact LOD we want to have in our visualization, and it is completely unaffected by the LOD we include in our view. Because of this, we kept the overall average of **`Employment`** in our calculation, despite introducing other dimensions into the view that would normally disaggregate our measure.

Tableau also offers the following two other types of LOD expressions. Let's take a quick look at what they do:

- `INCLUDE`: These expressions allow us to introduce an additional LOD into the view that is more granular than the one determined by the visualization itself. The `INCLUDE` expressions take into account the LOD that is included in the visualization, but always disaggregates data additionally by the dimension we specified in the `INCLUDE` expression itself.

- `EXCLUDE`: These expressions do the opposite of the `INCLUDE` expressions---they keep the data at the certain level of granularity that is higher than the one introduced by the view itself. This means that, when we create our visualization, Tableau will aggregate the measure according to the view, with the exception of the dimension that is specified in the `EXCLUDE` expression. This will result in the values repeating across multiple data points in our view because, instead of aggregating them across these data points, they are aggregated at a higher level.

Using custom geocoding

Tableau can recognize many locations as geographical values and assign geographical coordinates to them, allowing us to map them without actually knowing their exact coordinates. But, sometimes, we would like to use map locations that Tableau does not recognize as geographical data and can't generate longitude and latitude for.

In those cases, we can provide Tableau with the geographical coordinates ourselves using custom geocoding.

Getting ready

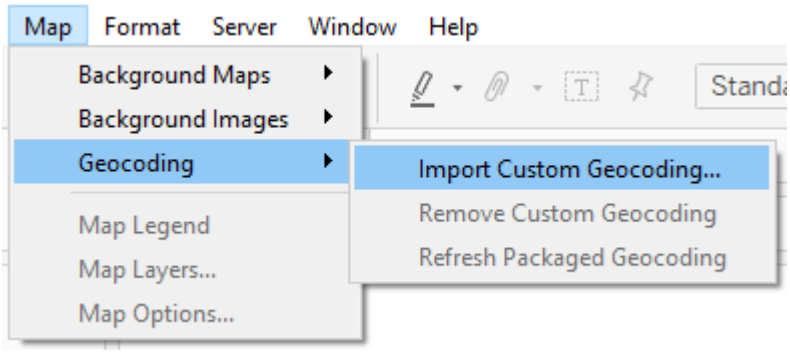
In this recipe, we will use two datasets: one we want to create our views from, and one that contains custom geocoding data. Make sure you have both the `Serbian_provinces_population_size.csv` and `Province_geocoding.csv` datasets saved to your device. Also, make sure the `Province_geocoding.csv` dataset is saved to a separate folder. Connect to the `Serbian_provinces_population_size.csv` dataset, and open a new blank worksheet.

Note

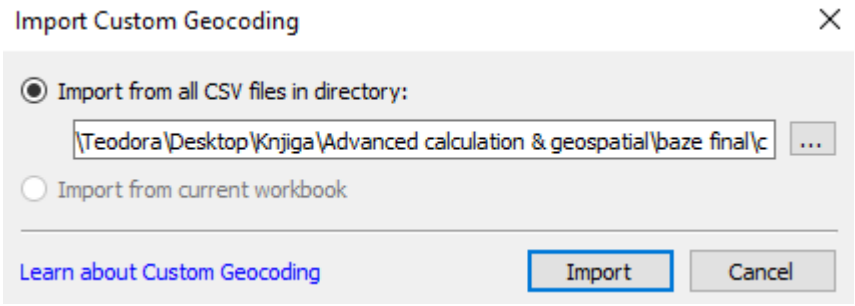
The dataset containing custom geocoding data (the longitude and latitude of the locations we would like to map) has to be stored in a text file, and in a separate folder.

How to do it...

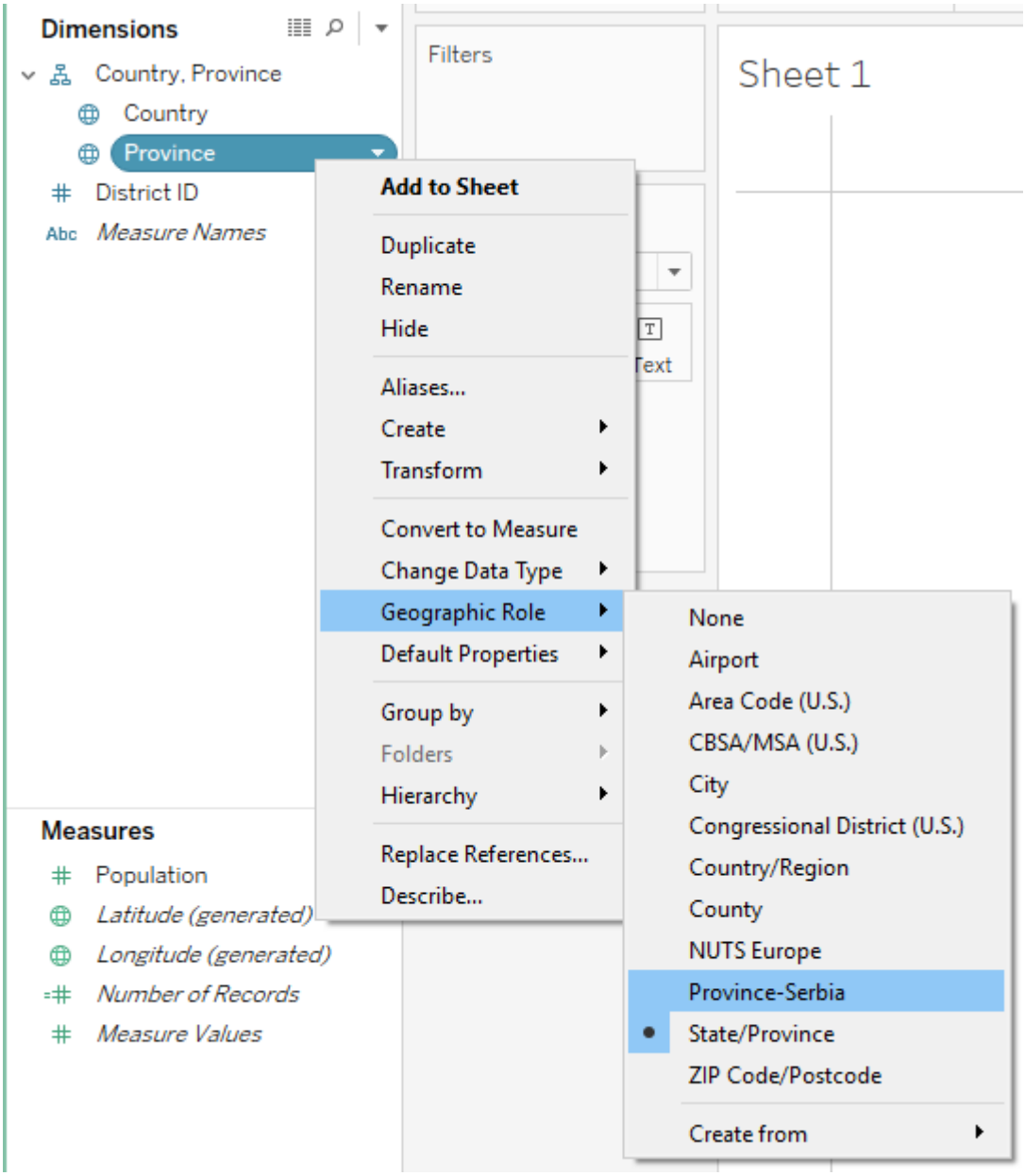
1. In the main menu toolbar, navigate to **`Map`**.
2. From the drop-down menu, navigate to **`Geocoding`** | **`Import Custom Geocoding`**... as shown in the following screenshot:



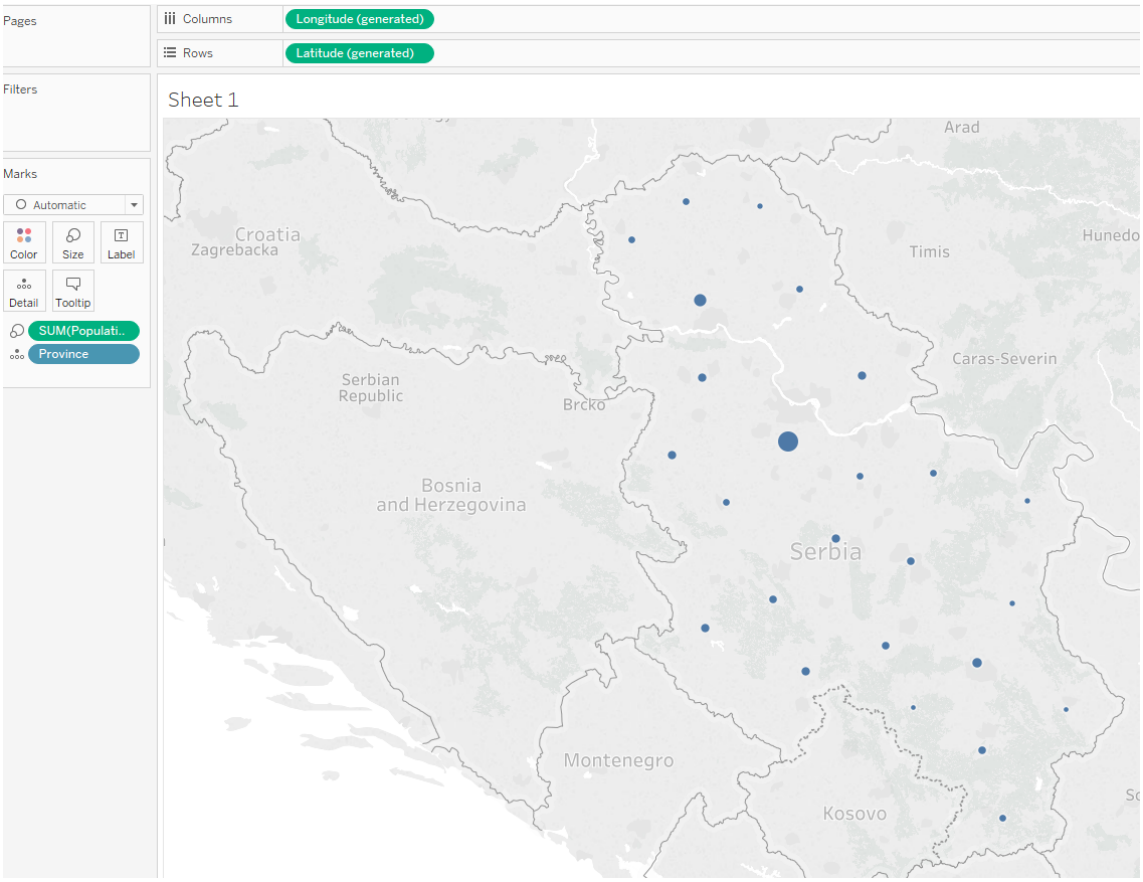
3. In the **`Import Custom Geocoding`** window, click on three dots (`...`) to open the directory where you saved `Province_geocoding.csv`:



4. In the **`Choose Source Folder`** window that opens, navigate to the folder in which you have saved the `Province_geocoding.csv` file with custom geocoding data, select it, and click **`Select Folder`**.
5. The path to the folder will now appear in the **`Import Custom Geocoding`** window. Click on **`Import`**, and wait while Tableau processes your request.
6. When the custom geocoding is imported, right-click on the ** `Province` **pill under **`Dimensions`**, and in the drop-down menu navigate to **`Geographical Role`** and choose **`Province-Serbia`**:



7. We have now set up our custom geocoding for **`Province-Serbia`**, which is also indicated by a small globe with a paper symbol ( ) that has appeared next to it. To use it in a map, just drag and drop **`Province`** from **`Dimensions`** into the workspace.
8. Finally, add **`Population`**, by dropping it from **`Measures`** onto **`Size`** in the **`Marks`** card as shown in the following screenshot:



How it works...

We used the `Province_geocoding.csv` dataset to set up custom geocoding for **`Province`**. The `Province_geocoding.csv` dataset contains longitudes and latitudes of each province. Once we have imported that data as geographical coordinates, we assign that geocoding to the dimension from the `Serbian_provinces_population_size.csv` dataset, **`Province`**. Since the names of the provinces are the same in both files, Tableau is able to make a connection between the **`Province`** dimension in the `Serbian_provinces_population_size.csv` file, and longitude and latitude from the `Province_geocoding.csv` file. Thanks to this, we can now map **`Province`** correctly.

There's more...

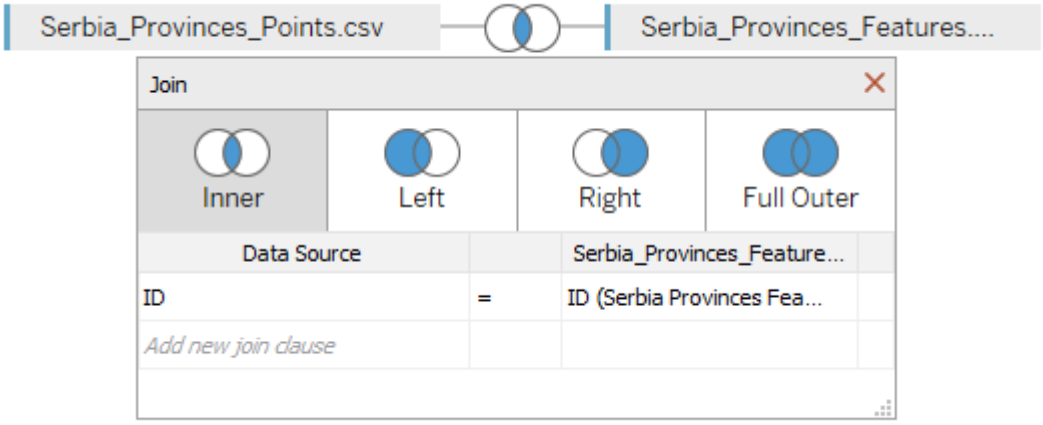
The same outcome could also be achieved with data blending. We could connect to both datasets, `Province_geocoding.csv` and `Serbian_provinces_population_size.csv`, and blend them by **`Province`** and **`Province-Serbia`**. This would allow us to use **`Longitude`** and **`Latitude`** from `Province_geocoding.csv` and combine it with **`Province`** from the `Serbian_provinces_population_size.csv` data source in our view. However, custom geocoding does have some advantages over this method. For example, once we import it, we can reuse it in other workbooks as well.

Using polygons for analytics

In the previous recipe, [*Using custom geocoding*], we have imported and used custom geocoding to map Serbian provincial centers. In this recipe, we will go a step further and map the region's borders in order to create a filled map. We will achieve this by using polygon mapping.

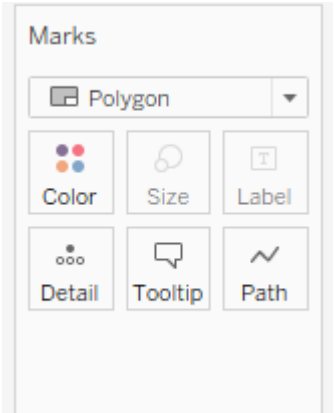
Getting ready

In this recipe, we will use two datasets: `Serbia_Provinces_Features.csv` and `Serbia_Provinces_Points.csv`. The `Serbia_Provinces_Features.csv` dataset contains data on Serbian provinces' populations, while `Serbia_Provinces_Points.csv` contains coordinates for mapping. Open them both, and make sure they are joined by **`ID`**. Double-click on the set intersection symbol to make sure the datasets are joined by **`ID`**:



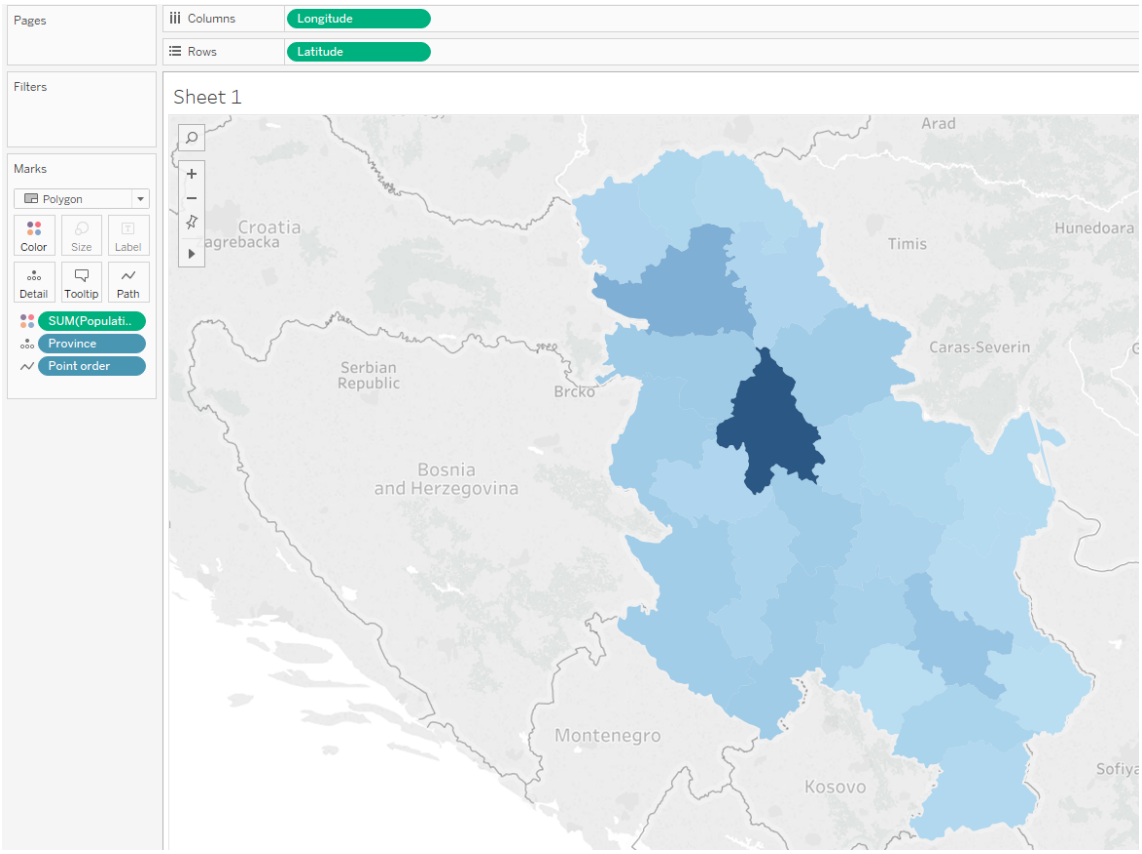
How to do it...

1. Drag and drop **`Longitude`** from **`Measures`**, the `Serbia_Provinces_Points.csv` table, into the **`Columns`** shelf.
2. Right-click on the **`Longitude`** pill in the **`Columns`** shelf, and select **`Dimension`**.
3. Drag and drop **`Latitude`** **from** **`Measures`**, the `Serbia_Provinces_Points.csv` table, into the **`Rows`** shelf.
4. Right-click on the **`Latitude`** pill in the **`Rows`** shelf, and select **`Dimension`**.
5. In the **`Marks`** card, use the drop-down menu to change mark type from **`Automatic`** to **`Polygon`**. You will notice a new button, **`Path`**, appearing in the **`Marks`** card, as shown in the following screenshot:



6. Right-click on the **`Point order`** field under **`Measures`** and select **`Convert to Dimension`**.
7. Drag and drop **`Point order`** from **`Dimensions`** onto **`Path`** in the **`Marks`** card.

8. Drag and drop `Province` from `Dimensions` onto `Detail` in the `Marks` card.
9. Finally, drag and drop `Population` from `Dimensions` to `Color`:



How it works...

Polygon maps are created by giving Tableau the coordinates of the shape we want to draw, and then connecting the dots by drawing a path between them. We have created a map with custom territories -- provinces---by giving Tableau the exact coordinates (longitude and latitude) of their borders, which were contained in the `Serbia_Provinces_Points.csv` dataset. The `Point order` dimension notifies Tableau in which order to connect the dots (coordinate points). Finally, placing `Province` in `Detail` completed our map.

There's more...

Polygon mapping allows us to draw any shape, not just geographical maps. As long as we know the [*x *]and [*y *]axis coordinates of the shape (for example, in pixels), and have them recorded in a file, we can draw any shape we like and use it in our visualizations. Mapping an image does require some patience, but getting a visualization of a customized shape that can be reused can be well worth it!