Calculate time-in-state for an existing Analytics view

04/04/2019 • 17 minutos para ler • Colaboradores

Neste artigo

Azure DevOps Services | Azure DevOps Server 2019

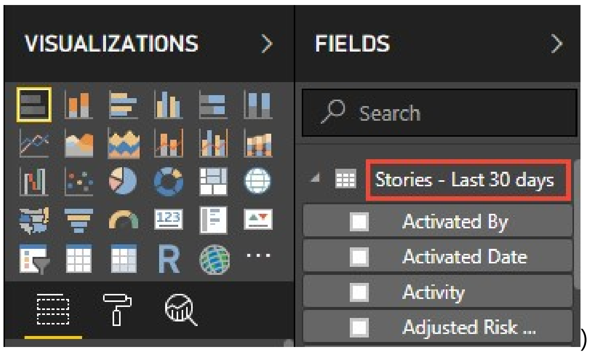
The time a work item spends in a specific state or series of states is an important aspect for understanding efficiency. The Analytics widgets, Cycle Time and Lead Time, provide some measures of time-in-state. However, these widgets may not have the level of detail that you want.

This article provides a series of recipes using DAX calculations to evaluate time spent by work items in any combination of states. Specifically, you'll learn how to add the following calculated columns and one measure and use them to generate various trend charts. (All fields are calculated columns except the first one listed.)

- *Work Items Count* (measure)
- *State Sort Order*
- *Date Previous*
- *Date Diff in Days*
- *Is Last Day in State*
- *State Time in Days*
- *State Previous*
- *State Changed*
- *State Flow*
- *State Change Count*
- *State Change Count - Last Proposed*
- *State Restart Time in Days*
- *State Time in Days - In Progress*

ⓘ **Importante**

- When adding a calculated column or measure per the examples shown in this article, replace *View Name* with the table name for the Analytics view. For example, replace *View Name* with *Stories - Last 30 days*.



)

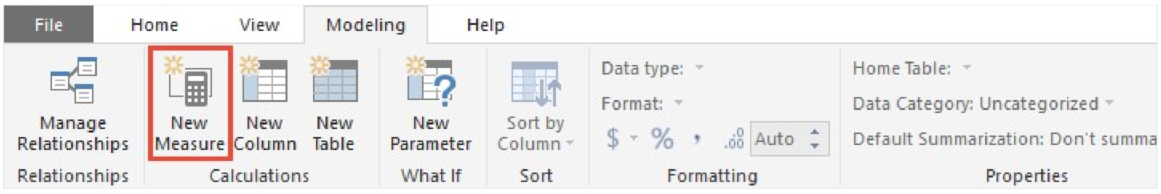
- Analytics views do not support intra-day revisions. These examples have the most precision when using a Daily interval for the Analytics view.
- All intra-day or intra-period (weekly/monthly) revisions are ignored by the calculations. This can result in unexpected results for specific scenarios like a work item showing no time "In Progress" when a work item is "In Progress" for less than a day.
- Power BI default aggregations are used whenever possible instead of building measures.
- Some calculations will include +0 to ensure that a numeric value is included for every row instead of BLANK. You may need to revise some of the calculated column definitions based on the workflow states used by your project. For example, if your project uses 'New", 'Active' and 'Closed' in place of 'Proposed', 'In Progress', and 'Completed'.

Add the *Work Items Count* measure

To simplify quickly generating reports, we designed Analytics views to work with default aggregations in Power BI. To illustrate the difference between a default aggregation and a measure, we start with a simple work item count measure.

1. Load the Power BI pbix file associated with your view in Power BI Desktop. For details, see Connect with Power BI Data Connector.

2. From the Modeling tab **Calculations** section of the ribbon, choose **New Measure**.



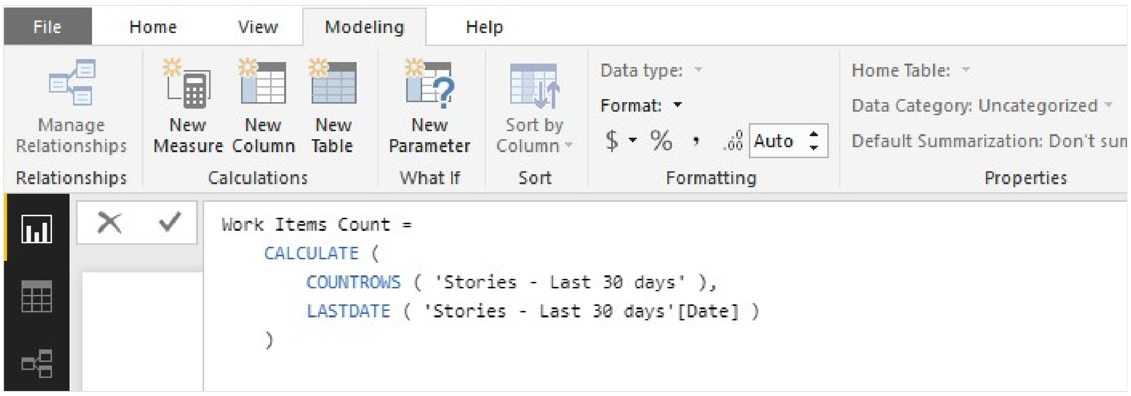
3. Replace the default text with the following code and then click the ✓ checkmark.

DAX

```
Work Items Count =
CALCULATE (
    COUNTROWS ( 'View Name' ),
    LASTDATE ( 'View Name'[Date] )
)
```

The *Work Items Count* measure uses the `CALCULATE`, `COUNTROWS`, and `LASTDATE` DAX functions which are described in more detail later in this article.

> ⓘ **Observação**
>
> Remember to replace *View Name* with the table name for the Analytics view. For example, here we replace *View Name* with *Stories - Last 30 days*.

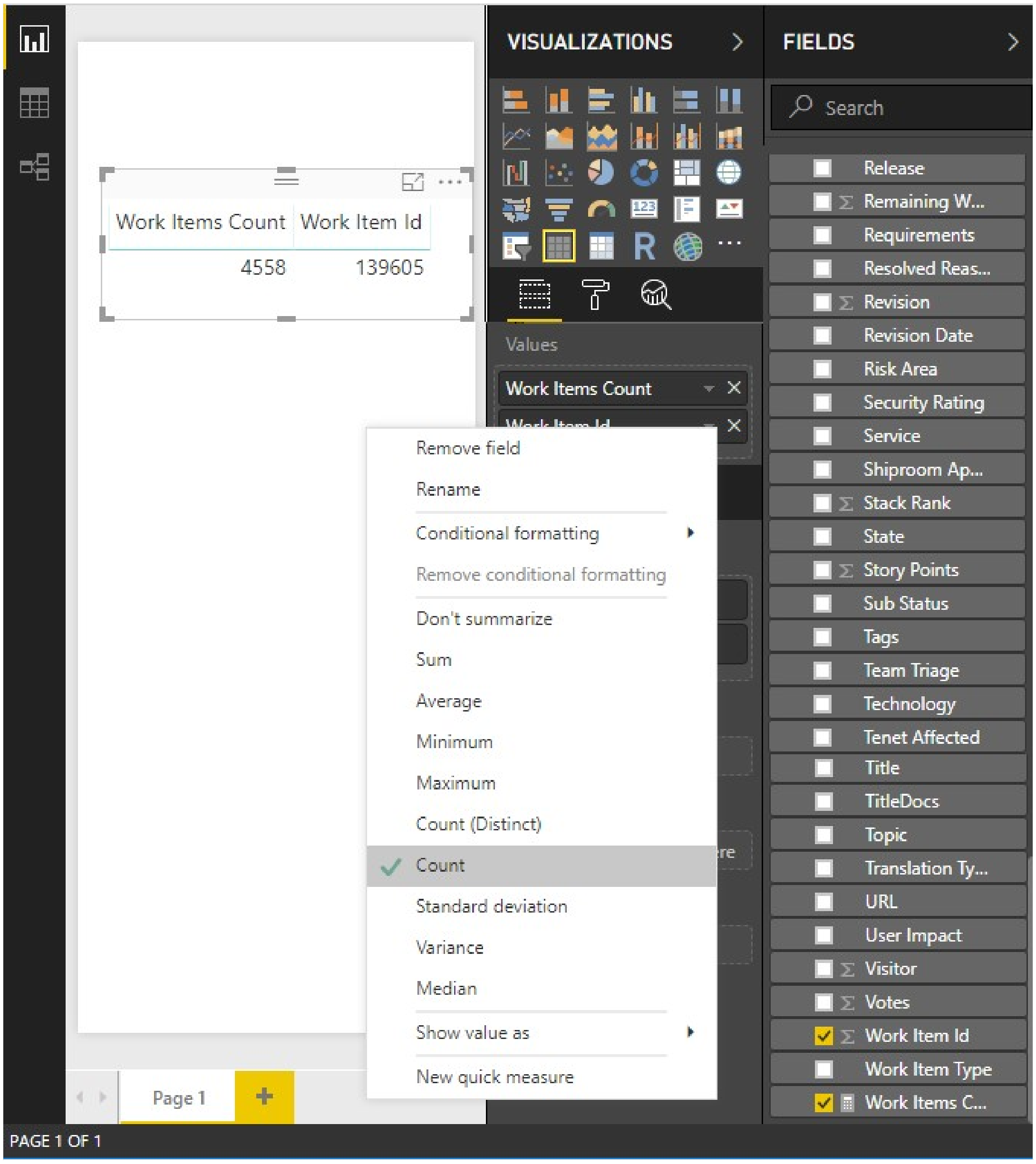


```
Work Items Count =
    CALCULATE (
        COUNTROWS ( 'Stories - Last 30 days' ),
        LASTDATE ( 'Stories - Last 30 days'[Date] )
    )
```

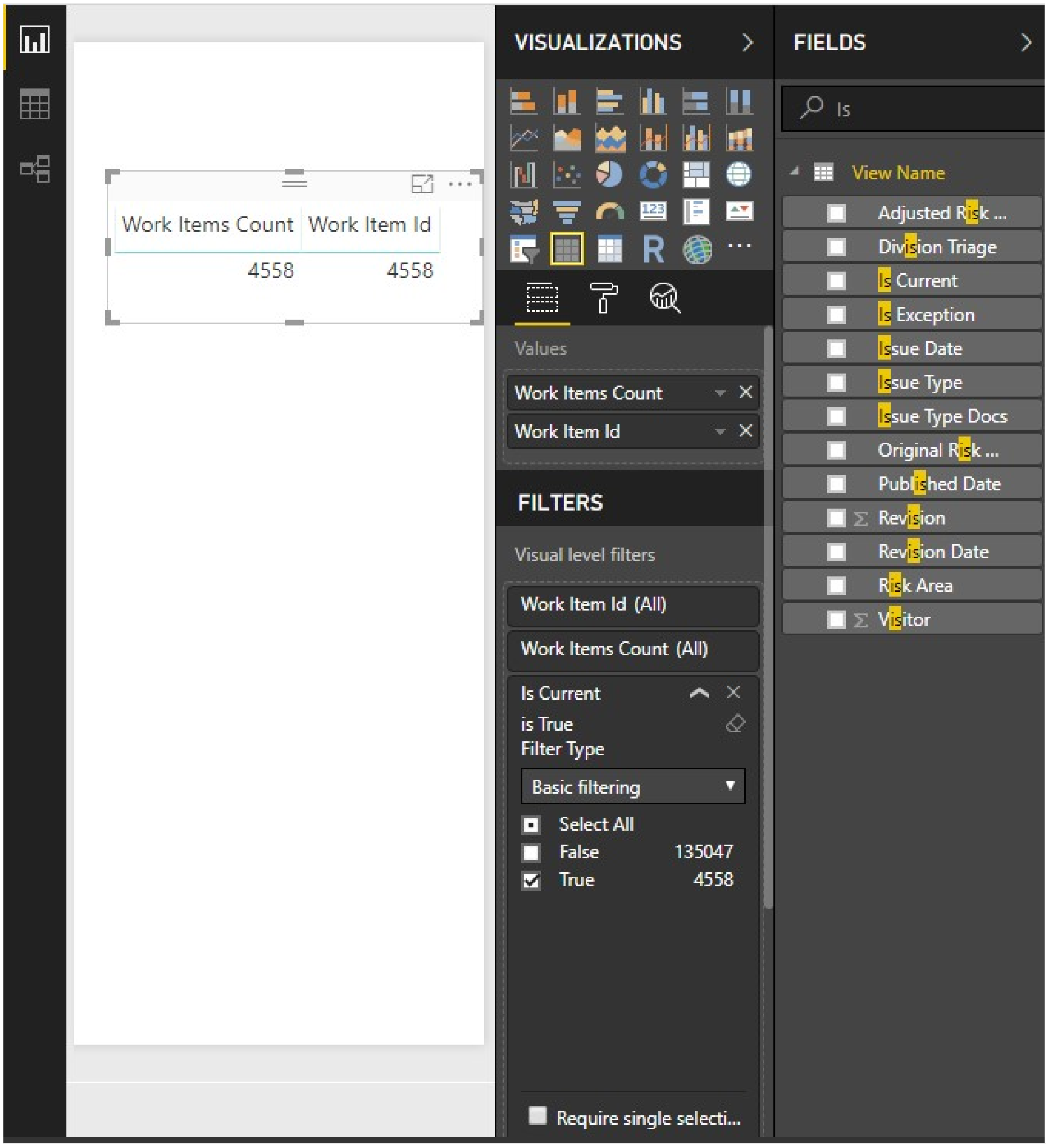
Understand the difference between a measure and a calculated column

Before describing the additional functions you'll create, you'll want to understand the difference between a measure and a calculated column. Specifically, a measure always evaluates the entire table where a calculated column is specific to a single row.

Compare the *Work Items Count* measure with the default count aggregation based on the *Work Item ID*:



To get the correct count using a default aggregation, you apply the filter *Is Current* equals 'True'. This pattern of applying filters to a default aggregation will be the basis for many of the examples provided in this article.



Work Items Count	Work Item Id
4558	4558

☐ False	135047
☑ True	4558

Add *State Sort Order*

By default, Power BI will show states sorted alphabetically in a visualization. This can be misleading when you want to visualize time in state and *Proposed* shows up after *In Progress*.

The following steps will help you resolve this issue.

1. Verify that the *State Category* field is included in the Analytics view. This field is included in all default shared views.

2. Choose **New Column** and replace the default text with the following code and then click the ✓ checkmark.

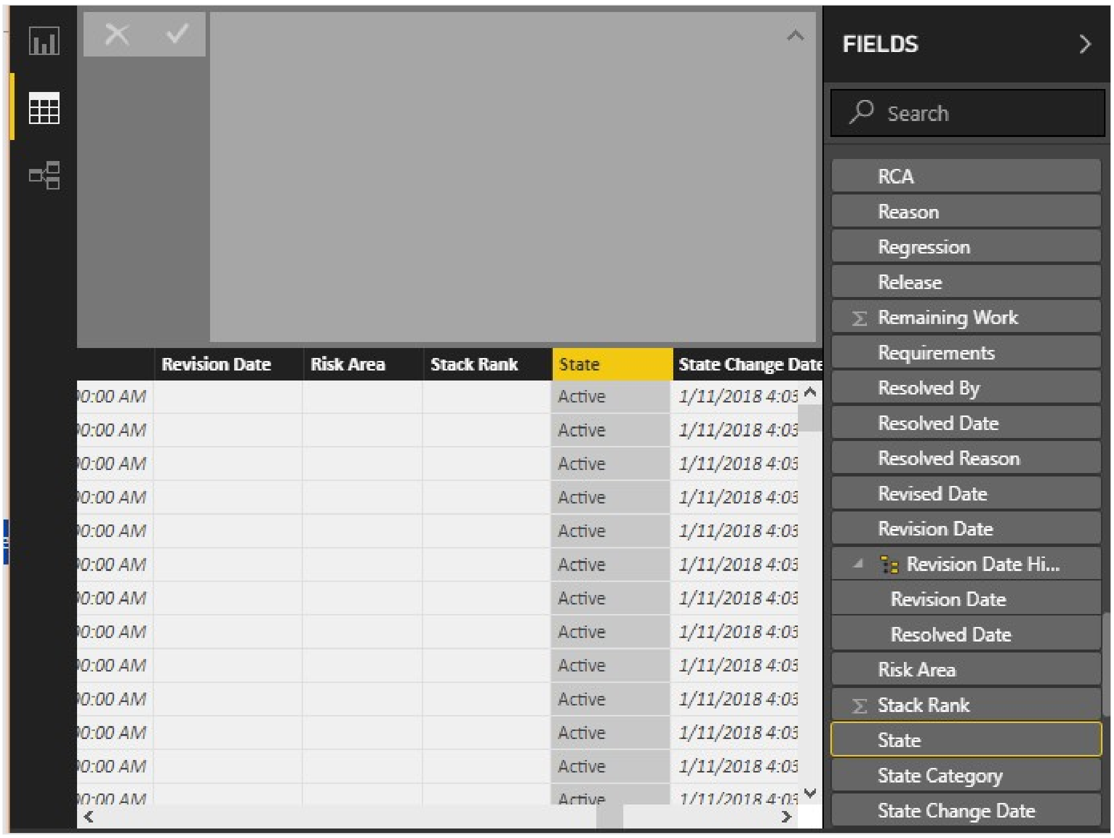
DAX Copiar

```
State Sort Order =
SWITCH (
    'View Name'[State Category],
    "Proposed", 1,
    "InProgress", 2,
    "Resolved", 3,
    4
)
```

ⓘ **Observação**

You may need to revise the definition if you need more granularity than *State Category* provides. *State Category* provides correct sorting across all work item types regardless of any *State* customizations.

3. Open the **Data** view and select the *State* field.



	Revision Date	Risk Area	Stack Rank	State	State Change Date
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03
0:00 AM				Active	1/11/2018 4:03

4. From the **Modeling** tab, choose **Sort by Column** and then select the *State Sort Order* field.

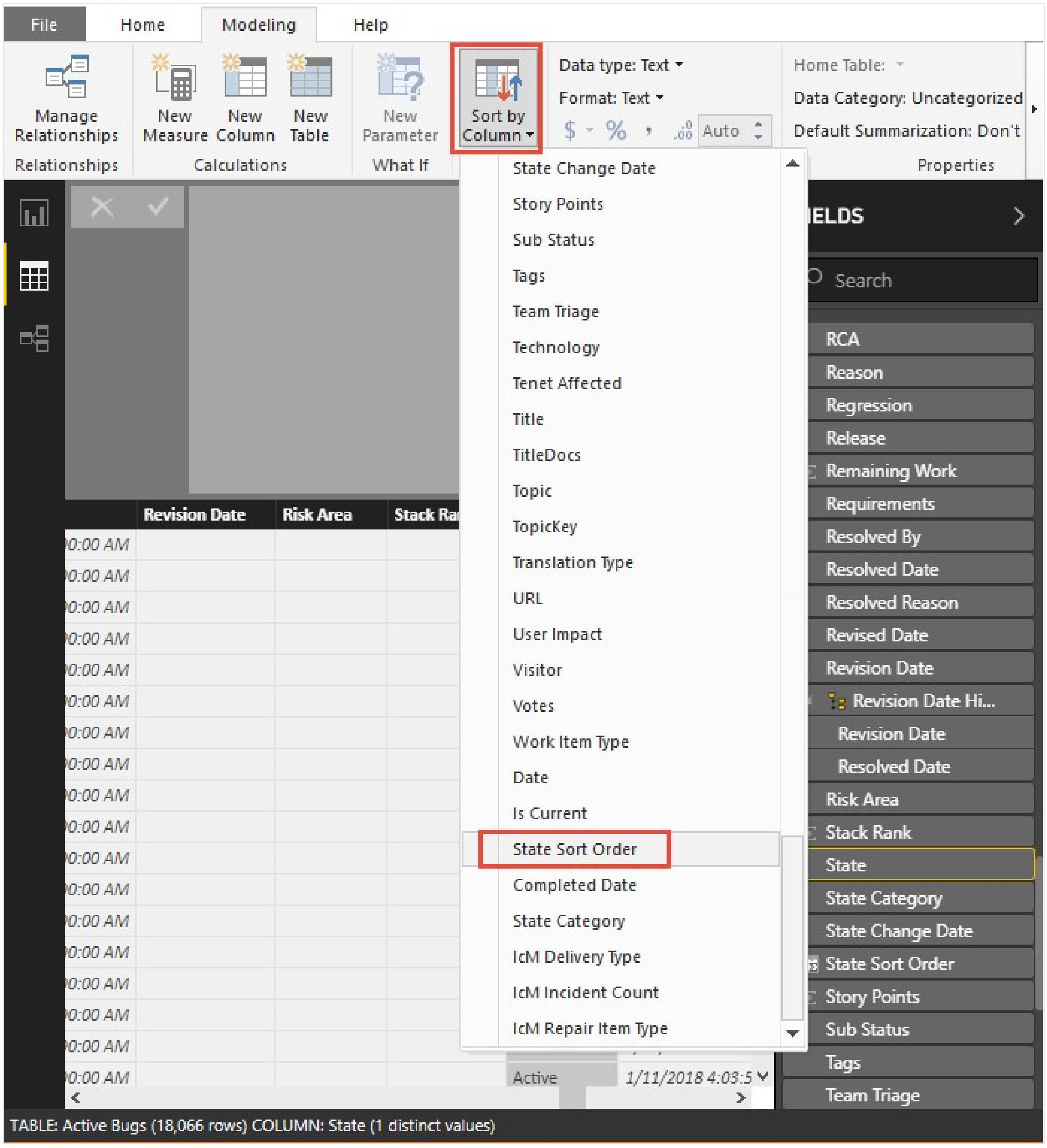


TABLE: Active Bugs (18,066 rows) COLUMN: State (1 distinct values)

Add *Date Previous*

The next step for calculating time-in-state requires mapping the previous interval (day, week, month) for each row of data in the dataset. This is a very simple calculation using a calculated column. Typically, you would define this column as shown.

DAX

Copiar

```
Date Previous  =
PREVIOUSDAY ( 'View Name'[Date] )
```

However, this approach has two main problems:

- It works only for daily periods.
- It does not handle gaps in the data. For example, if a work item is moved between projects.

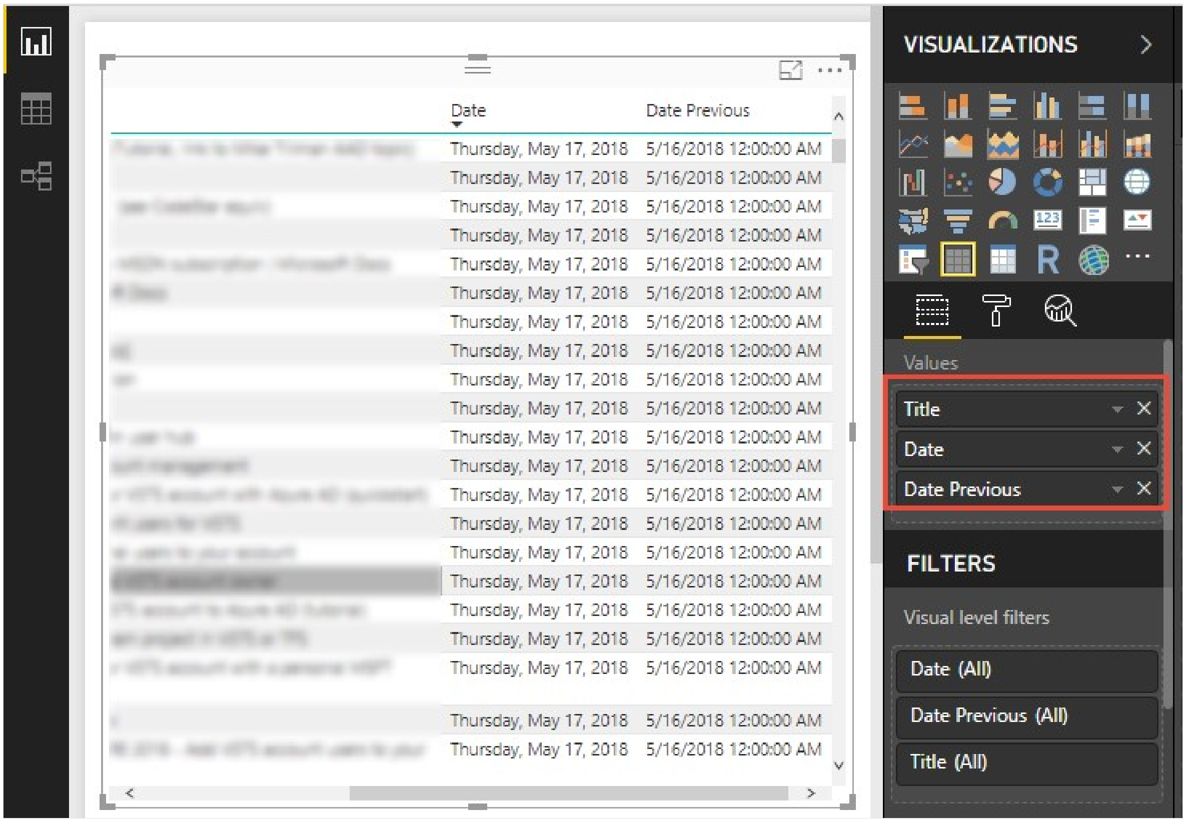
To resolve these problems, the calculated column should find the previous day by scanning the *Date* field.

To add the *Date Previous* calculated column, from the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX Copiar

```
Date Previous =
CALCULATE (
    MAX ( 'View Name'[Date] ),
        ALLEXCEPT ( 'View Name', 'View Name'[Work Item Id] ),
        'View Name'[Date] < EARLIER ( 'View Name'[Date] )
)
```

This calculated column uses three DAX functions, `MAX`, `ALLEXCEPT`, and `EARLIER`, described in more detail later in this article. Keep in mind that because this is a calculated column it is run for each and every row in the table and each time it is run it has the context of that specific row.



	Date	Date Previous
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM
[illegible]	Thursday, May 17, 2018	5/16/2018 12:00:00 AM

Dica

From the context menu for the *Date* and *Previous Date* fields, choose **Date** (instead of **Date Hierarchy**) to see a single date for these fields.

Add *Date Diff in Days*

Date Previous calculates the difference between the previous and current date for each row. With *Date Diff in Days*, we'll calculate a count of days between each of those periods. For most rows in a daily snapshot, the value will equal 1. However, for many work items which have gaps in the dataset, the value will be larger than 1.

Importante

Requires that you have added the *Date Previous* calculated column to the table.

It is important to consider the first day of the dataset where *Date Previous* is blank. In this example we give that row a standard value of 1 to keep the calculation consistent.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX Copiar

```
Date Diff in Days =
IF (
    ISBLANK ( 'View Name'[Date Previous] ),
    1,
    DATEDIFF (
        'View Name'[Date Previous],
        'View Name'[Date],
        DAY
    )
)
```

This calculated column uses the `ISBLANK` and `DATEDIFF` DAX functions described later in this article.

Add *Is Last Day in State*

In this next step, we calculate if a given row represents the last day a specific work item was in a state. This supports default aggregations in Power BI with the next column we'll add, the *State Time in Days*.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX Copiar

```
Is Last Day in State =
ISBLANK (CALCULATE (
    COUNTROWS ( 'View Name' ),
        ALLEXCEPT ( 'View Name', 'View Name'[Work Item Id] ),
        'View Name'[Date] > EARLIER ( 'View Name'[Date] ),
        'View Name'[State] = EARLIER ( 'View Name'[State] )
))
```

Add *State Time in Days*

The time that a work item spent in a specific state can now be calculated by summing the *Date Diff in Days* for each work item. This calculation will include all of the time spent in a specific state even if it has switched between states multiple times. It is possible to evaluate each row as a trend using *Date* or the latest information by using *Is Last Day In State*.

ⓘ **Importante**

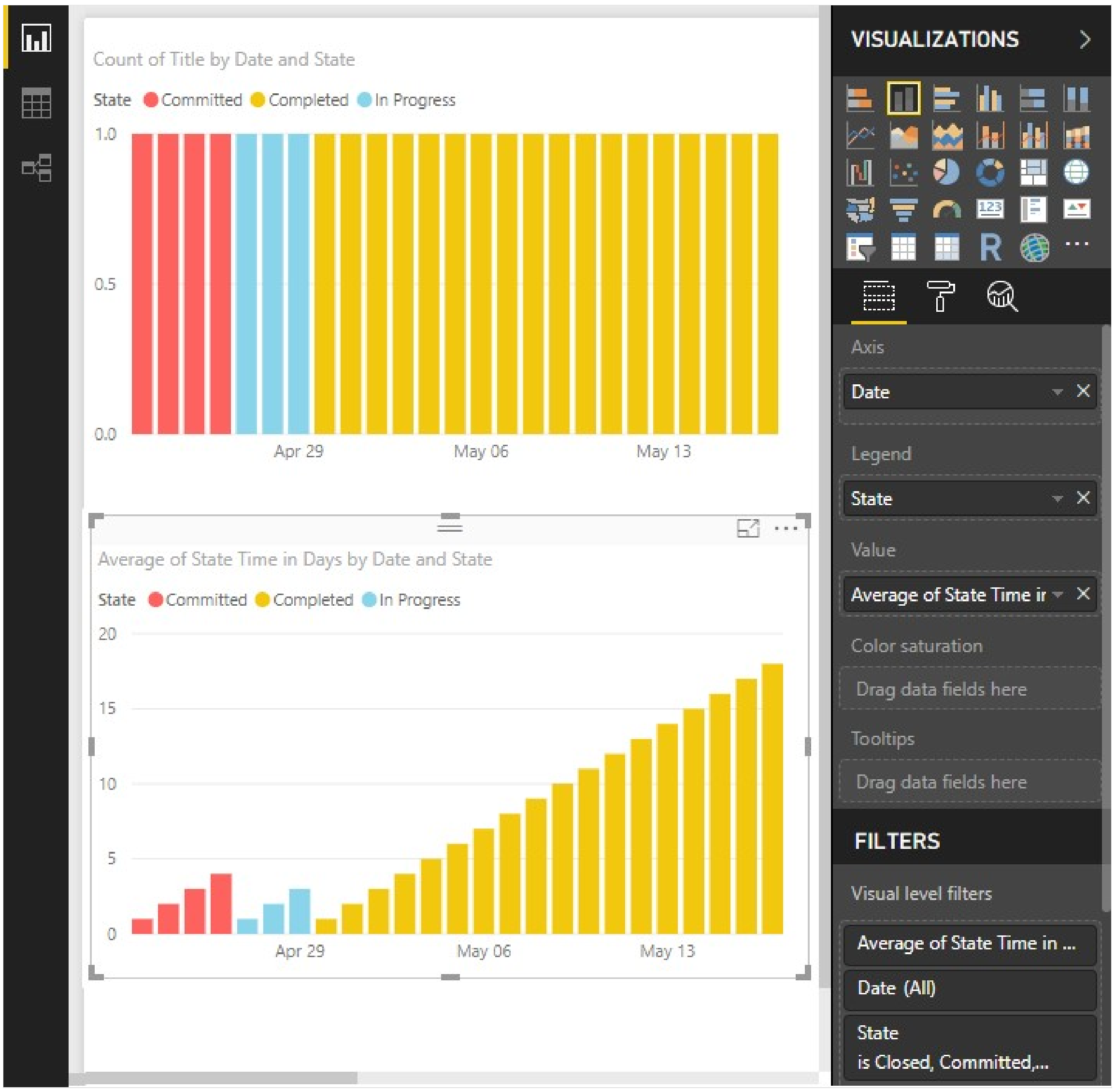
Requires that you have added the *Date Diff in Days* and *Is Last Day in State* calculated columns to the table.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX Copiar

```
State Time in Days =
CALCULATE (
    SUM ( 'View Name'[Date Diff in Days] ),
    ALLEXCEPT ( 'View Name', 'View Name'[Work Item Id] ),
    'View Name'[Date] <= EARLIER ( 'View Name'[Date] ),
    'View Name'[State] = EARLIER ( 'View Name'[State] )
) + 0
```

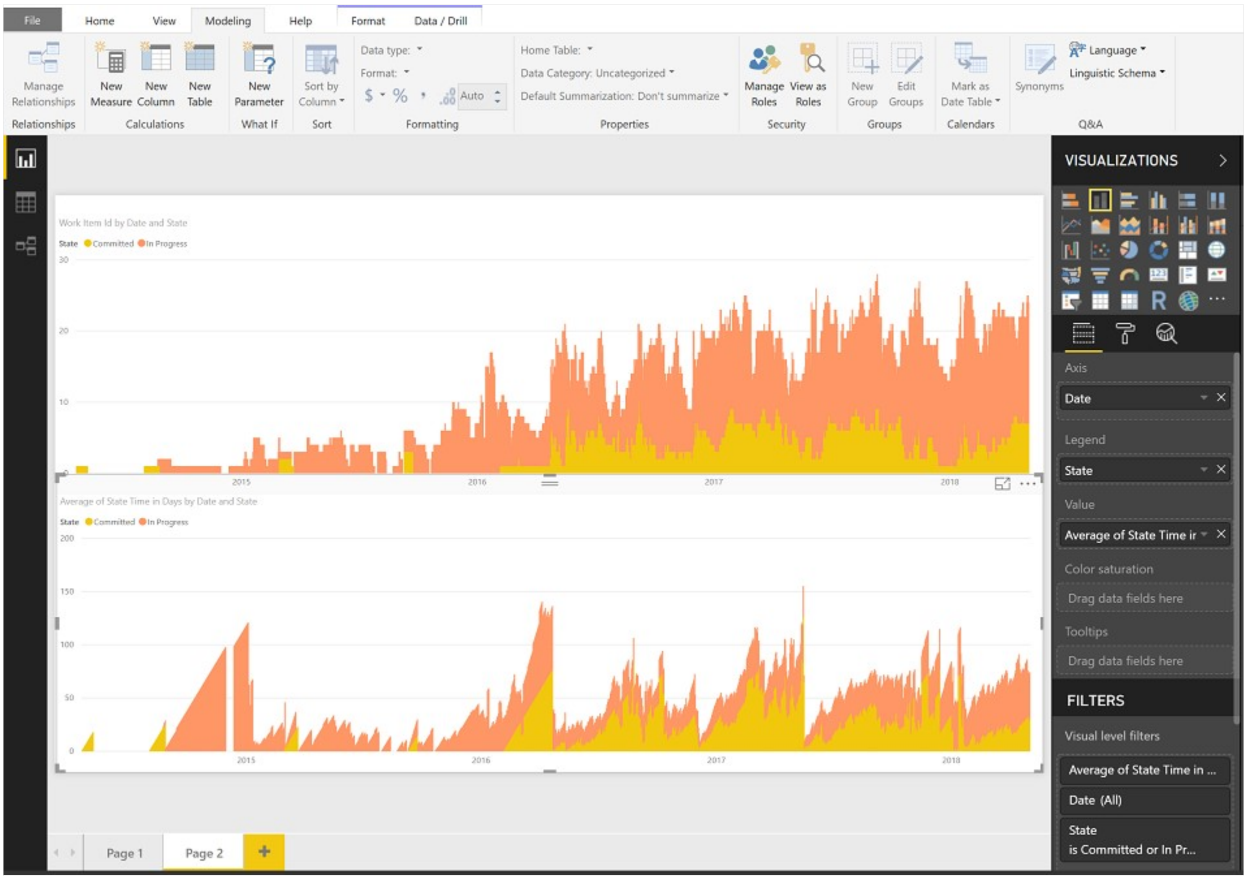
To demonstrate the *State Time in Days* column the report below includes a chart (top) showing a single work item and the state it was in on each day. The second chart (bottom) shows the time that this work item has spent in each state over time. Notice that the state increases by one each day until it moves to the next state.



Create a trend chart based on *State Time in Days*

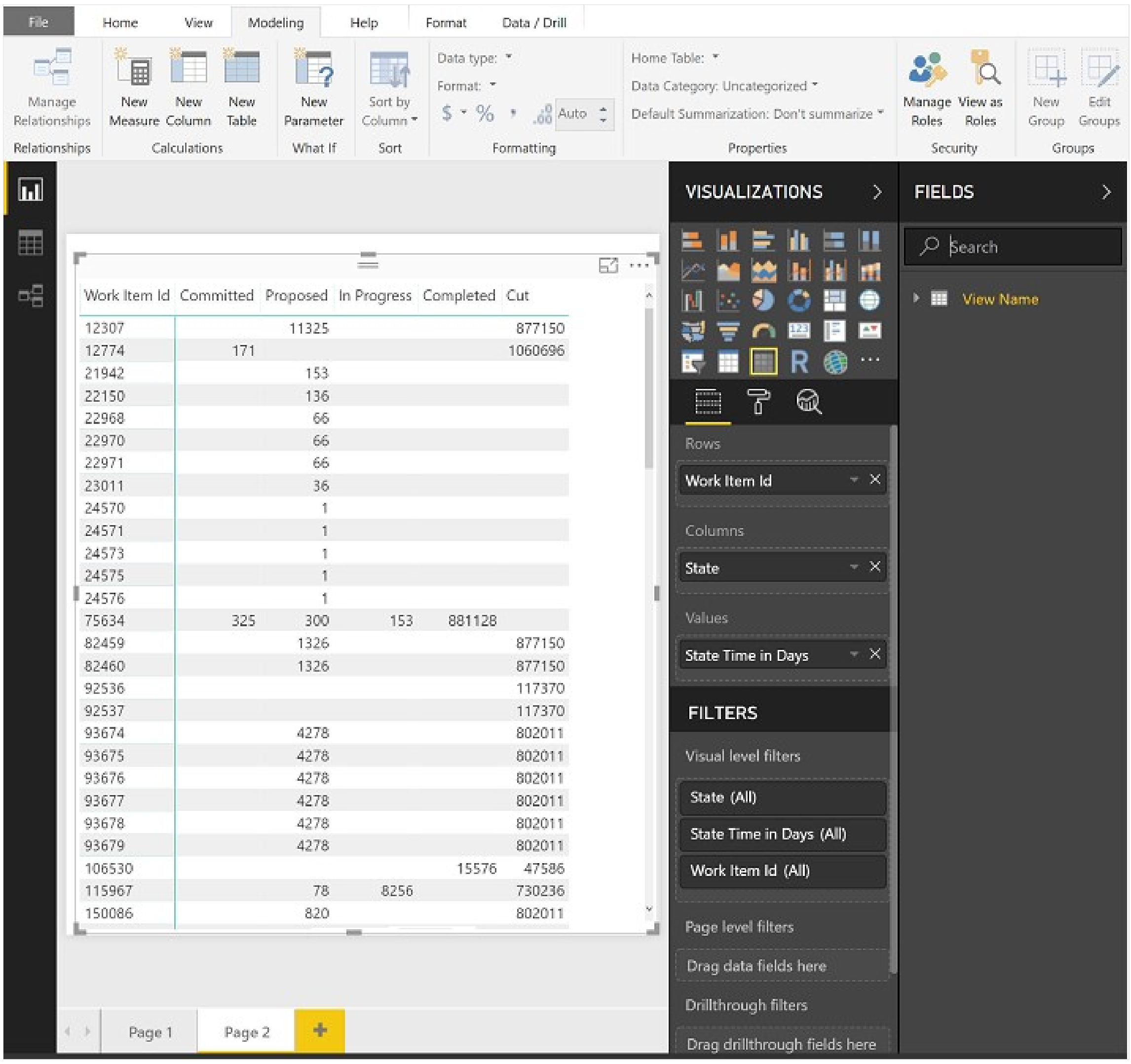
You can create a trend chart of an average time in state across all work items that were in that state on a given day using the *State Time in Days* calculated column in addition to *Date*.

To demonstrate, the same visualization used above are updated to show an average across all work items. This representation can help build understanding of the amount of work in progress and how it impacts the overall time in state for the teams. This team has a pattern of committing to work that they aren't ready to start work on. (Note that the data for the following report is based on different data than the previous report.)



Add *State Time in Days - Latest* (Is Last Day In State)

When evaluating time-in-state for each work item in a table or when filtered by a field like *Area Path*, do not use the *State Time in Days* column in an aggregation. The aggregation will use the value for every day the work item was in the state. For example, if a work item was *In Progress* on Monday and moved to *Completed* on Thursday, the time-in-state is 3 days, but the sum of *State Time in Days* column is 6 days (1+2+3) which is clearly incorrect.



Work Item Id	Committed	Proposed	In Progress	Completed	Cut
12307		11325			877150
12774	171				1060696
21942		153			
22150		136			
22968		66			
22970		66			
22971		66			
23011		36			
24570		1			
24571		1			
24573		1			
24575		1			
24576		1			
75634	325	300	153	881128	
82459		1326			877150
82460		1326			877150
92536					117370
92537					117370
93674		4278			802011
93675		4278			802011
93676		4278			802011
93677		4278			802011
93678		4278			802011
93679		4278			802011
106530				15576	47586
115967		78	8256		730236
150086		820			802011

Rows: Work Item Id

Columns: State

Values: State Time in Days

FILTERS

Visual level filters

State (All)

State Time in Days (All)

Work Item Id (All)

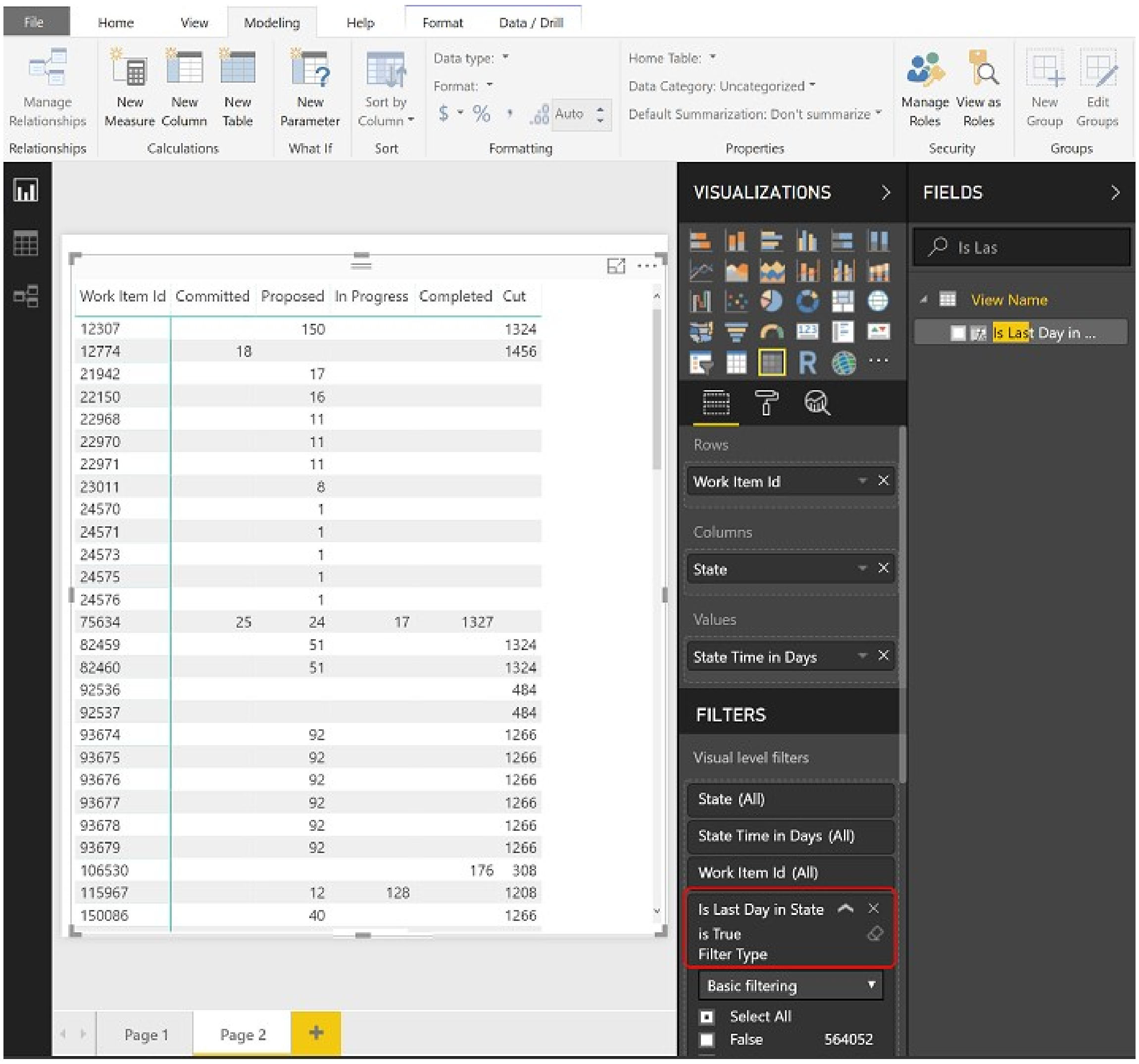
Page level filters

Drag data fields here

Drillthrough filters

Drag drillthrough fields here

To resolve this issue, use *State Time in Days* and apply the filter *Is Last Day In State* equals 'True'. This eliminates all the historical data necessary for a trend and focuses instead on just the latest value for each state.



Work Item Id	Committed	Proposed	In Progress	Completed	Cut
12307		150			1324
12774	18				1456
21942		17			
22150		16			
22968		11			
22970		11			
22971		11			
23011		8			
24570		1			
24571		1			
24573		1			
24575		1			
24576		1			
75634	25	24	17	1327	
82459		51			1324
82460		51			1324
92536					484
92537					484
93674		92			1266
93675		92			1266
93676		92			1266
93677		92			1266
93678		92			1266
93679		92			1266
106530				176	308
115967		12	128		1208
150086		40			1266

Data type:
Format:
Home Table:
Data Category: Uncategorized
Default Summarization: Don't summarize

Rows: Work Item Id
Columns: State
Values: State Time in Days

FILTERS
Visual level filters
State (All)
State Time in Days (All)
Work Item Id (All)
Is Last Day in State
is True
Filter Type
Basic filtering
Select All
False 564052

Add *State Time in Days - In Progress*

In the examples above *State Time in Days* for a given work item is only counted during the period of time when the work item was in that specific state. If your goal is to have the time-in-state for a given work item count towards an average continuously, You must change the calculation. For example, if we want to track the "In Progress" state, we add the *State Time in Days - In Progress* calculated column.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX | Copiar

```
State Time in Days - In Progress =
CALCULATE (
    SUM ( 'View Name'[Date Diff in Days] ),
    ALLEXCEPT ( 'View Name', 'View Name'[Work Item Id] ),
   'View Name'[Date] <= EARLIER('View Name'[Date]),
   'View Name'[State] = "In Progress"
) + 0
```

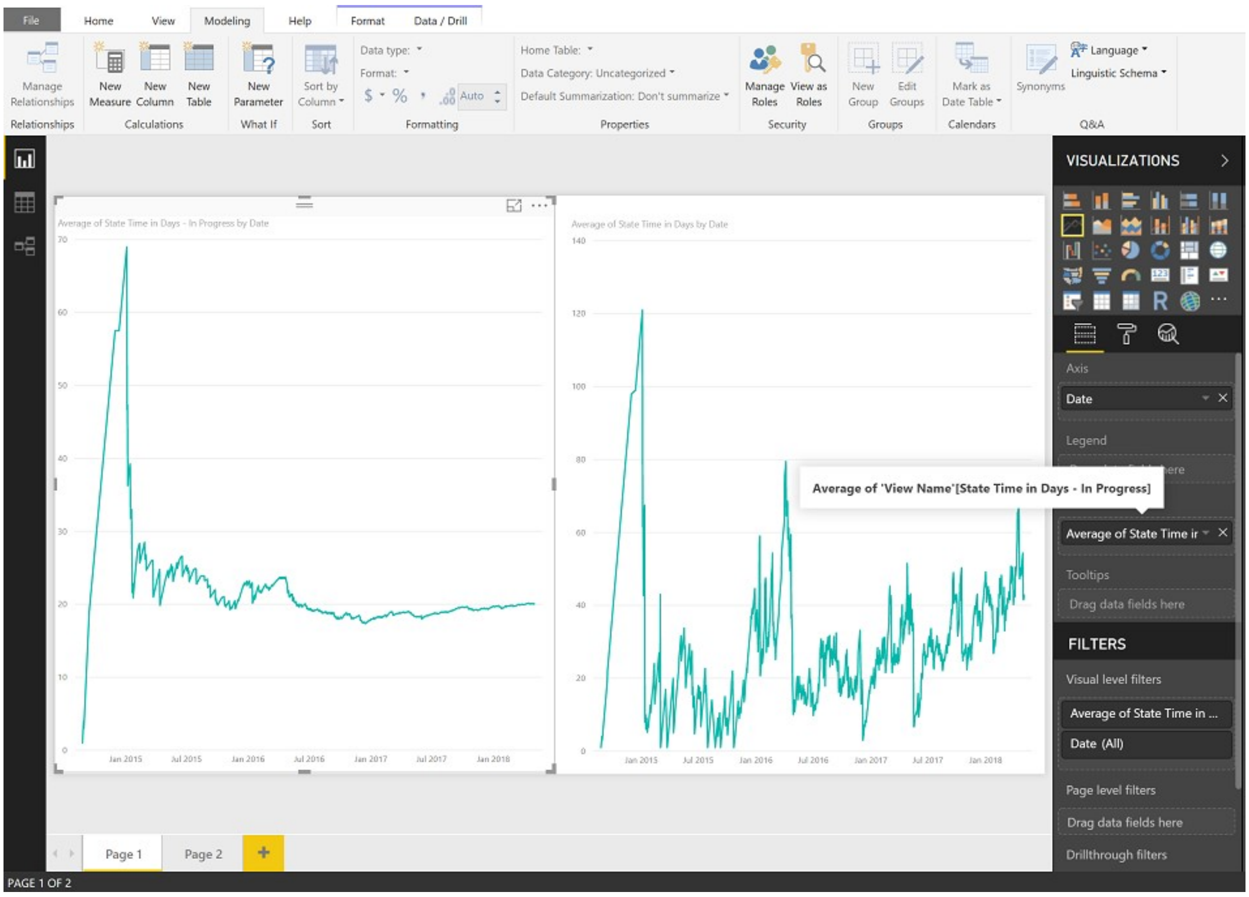
ⓘ **Observação**

You may need to revise the definition based on the workflow states used by your project. For example, if your project uses 'Active' in place of 'In Progress'.

ⓘ **Observação**

You may need to revise the definition based on the workflow states used by your project. For example, the project used in the examples in this article use the 'In Progress' workflow state, however, Agile, Scrum, and CMMI processes typically use the 'Active' or 'Committed' states to represent work in progress. For an overview, see [Workflow states and state categories](#).

The following image shows the impact of considering all time-in-state for every existing work item (shown left) versus only those work items in a specific state on a given day (shown right).



State Time in Days trend of multiple states

Analyzing performance across multiple states is also possible using the "Continuous" pattern. However, this approach only works with a trend chart.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

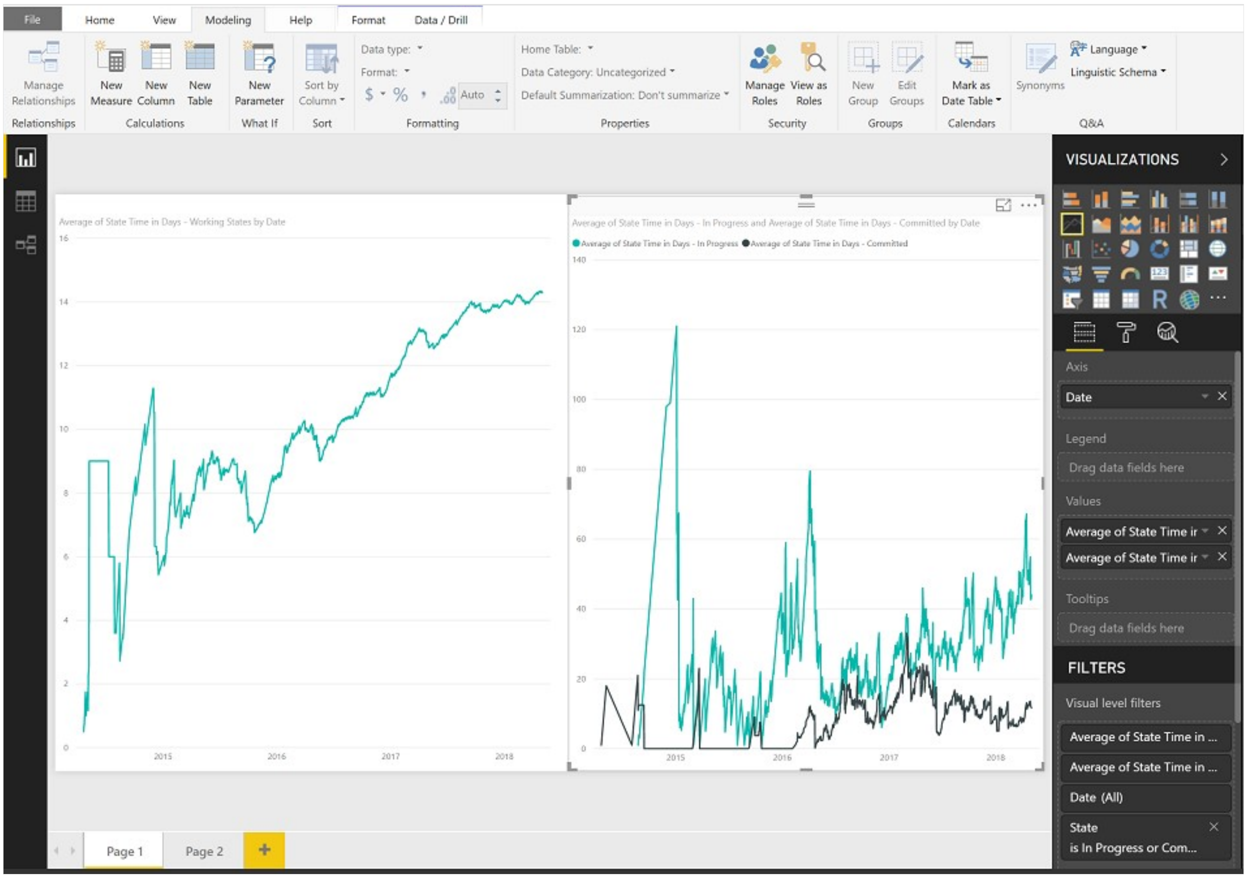
DAX Copiar

```
State Time in Days - Working States =
CALCULATE (
    SUM ( 'View Name'[Date Diff in Days] ),
    ALLEXCEPT ( 'View Name', 'View Name'[Work Item Id] ),
   'View Name'[Date] <= EARLIER('View Name'[Date]),
   'View Name'[State] IN { "Committed", "In Progress" }
) + 0
```

ⓘ **Observação**

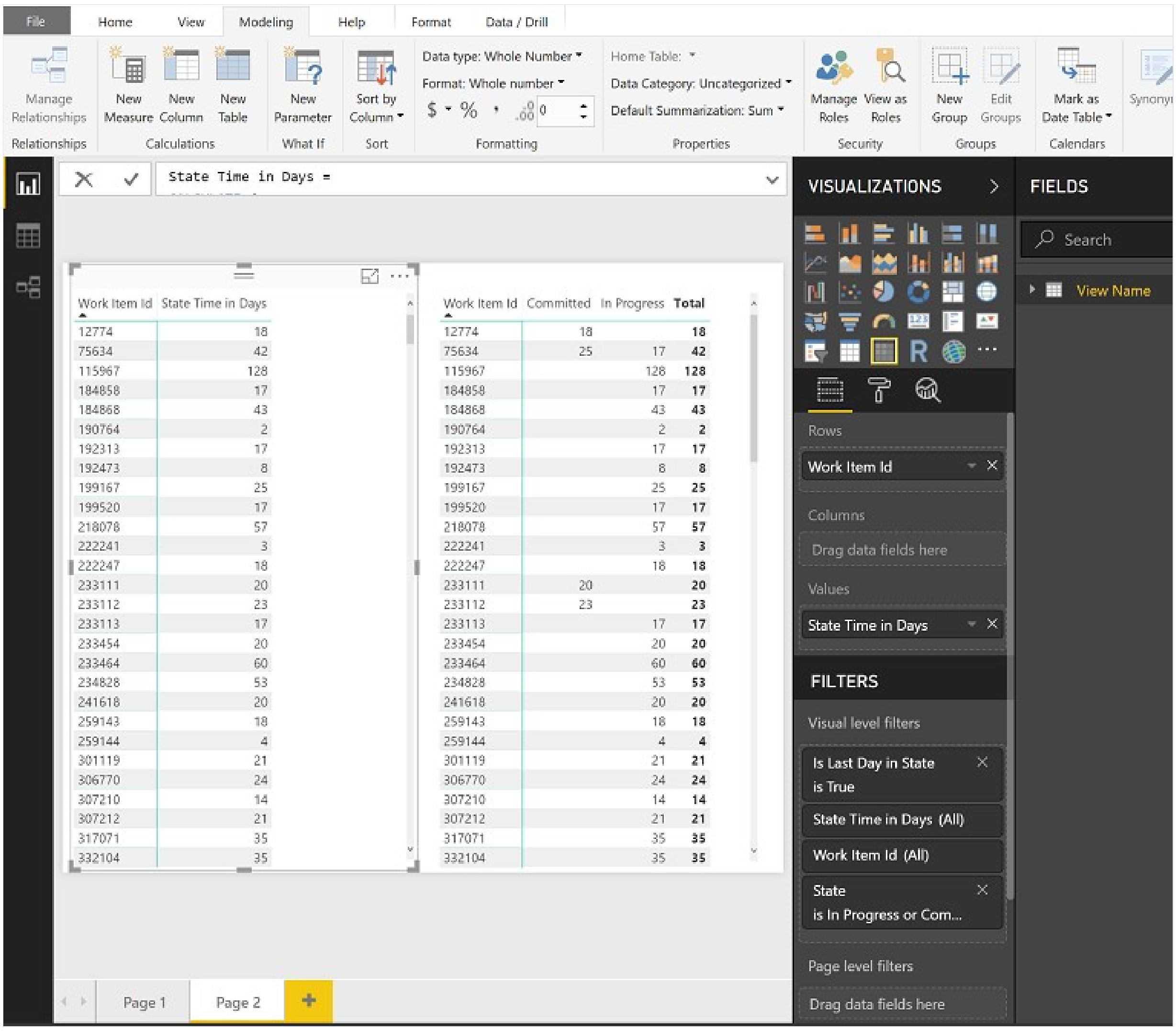
You may need to revise the definition based on the workflow states used by your project. For example, if your project uses 'Active' in place of 'Committed' or 'Proposed'.

The chart of the left shows combined average while the right hand side shows each individual state.



Get the State Time in Days- Latest for multiple states

You use the *State Time in Days- Latest* calculated column when creating a trend. Using a filter on the states, the *State Time in Days* column and *Is Last Day in State* provides a simple way to get the total time any work item or group of work items spent in a set of states.



Data type: Whole Number
Format: Whole number
Home Table:
Data Category: Uncategorized
Default Summarization: Sum

```
State Time in Days =
```

Work Item Id	State Time in Days
12774	18
75634	42
115967	128
184858	17
184868	43
190764	2
192313	17
192473	8
199167	25
199520	17
218078	57
222241	3
222247	18
233111	20
233112	23
233113	17
233454	20
233464	60
234828	53
241618	20
259143	18
259144	4
301119	21
306770	24
307210	14
307212	21
317071	35
332104	35

Work Item Id	Committed	In Progress	**Total**
12774	18		**18**
75634	25	17	**42**
115967		128	**128**
184858		17	**17**
184868		43	**43**
190764		2	**2**
192313		17	**17**
192473		8	**8**
199167		25	**25**
199520		17	**17**
218078		57	**57**
222241		3	**3**
222247		18	**18**
233111	20		**20**
233112	23		**23**
233113		17	**17**
233454		20	**20**
233464		60	**60**
234828		53	**53**
241618		20	**20**
259143		18	**18**
259144		4	**4**
301119		21	**21**
306770		24	**24**
307210		14	**14**
307212		21	**21**
317071		35	**35**
332104		35	**35**

Rows
Work Item Id
Columns
Drag data fields here
Values
State Time in Days
FILTERS
Visual level filters
Is Last Day in State is True
State Time in Days (All)
Work Item Id (All)
State is In Progress or Com...
Page level filters
Drag data fields here

Add *State Previous*

The *Date Previous* calculated column can also be used to lookup past values such as previous state for each work item.

ⓘ **Importante**

Requires that you have added the [***Date Previous*** **calculated column**] to the table.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX Copiar

```
State Previous =
LOOKUPVALUE (
    'View Name'[State],
    'View Name'[Work Item Id], 'View Name'[Work Item Id],
    'View Name'[Date], 'View Name'[Date Previous]
)
```

This calculated column uses the `LOOKUPVALUE`, described later in this article.

The first `LOOKUPVALUE` parameter, `'View Name'[State]`, specifies to return the value of [State].

The next parameter, `'View Name'[Work Item Id], 'View Name'[Work Item Id]`, specifies that only rows with a matching work item id as the current row should be considered.

And, the last parameter, `'View Name'[Date], 'View Name'[Date Previous]`, specifies that the date of the row being returned must have a [Date] that matches the [Previous Date] of the current row. In a snapshot only one row can meet this criteria.

Add *State Changed*

Using the *State Previous* column, we can flag the rows for each work item where a state transition has occurred. The *Stage Changed* calculated column you'll add has two special considerations:

- Blank values of *State Previous* will be set to the *Created Date* of the work item
- Creation of a work item is considered a state transition

ⓘ **Importante**

Requires that you have added the ***State Previous*** calculated column to the table.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX Copiar

```
State Changed =
IF (
    ISBLANK ( 'View Name'[State Previous] ),
    'View Name'[Created Date].[Date] = 'View Name'[Date],
    'View Name'[State Previous] <> 'View Name'[State]
)
```

Add *State Flow*

With *State Previous* and *State Changed* calculated columns, you can create a column that will help illustrate the State Flow for a given work item. Creating this column is optional for the purposes of this article.

ⓘ **Importante**

Requires that you have added the State Previous and ***State Changed*** calculated columns to the table.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX Copiar

```
State Flow =
IF([State Changed], [State Previous], [State]) & " ž¡ " & [State]
```

Add *State Change Count*

As we move into the more complicated measures, we need to have a representation of the total number of state changes to compare the rows of a data for a given work item. We achieve this by adding a *State Change Count* calculated column.

> ⓘ **Importante**
>
> Requires that you have added the ***State Changed*** calculated column to the table.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX   Copiar

```
State Change Count =
CALCULATE (
    COUNTROWS ( 'View Name' ),
    ALLEXCEPT ( 'View Name', 'View Name'[Work Item Id] ),
    'View Name'[Date] <= EARLIER ( 'View Name'[Date] ),
    'View Name'[State Changed]
) + 0
```

Add *State Change Count - Last Proposed* and *State Restart Time in Days*

Calculating restarts, *State Restart Time in Days*, is a fairly complex calculation. The first step is to find the last time a work item was in a proposed state. Add the *State Change Count - Last Proposed* calculated column.

> ⓘ **Observação**
>
> You may need to revise the following definitions based on the workflow states used by your project. For example, if your project uses 'New' in place of 'Proposed'.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX   Copiar

```
State Change Count - Last Proposed =
CALCULATE (
    MAX ( 'View Name'[State Change Count] ),
    ALLEXCEPT ( 'View Name', 'View Name'[Work Item Id] ),
    'View Name'[Date] <= EARLIER ( 'View Name'[Date] ),
    'View Name'[State] = "Proposed"
)
```

Then, look further back to the past and see if there were some active states before this proposed state. Lastly, sum up all the days when work item was in active state before the last proposed.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX   Copiar

```
State Restart Time in Days =
CALCULATE (
    SUM ( 'View Name'[Date Diff in Days] ),
    ALLEXCEPT ( 'View Name', 'View Name'[Work Item Id] ),
    'View Name'[Date] <= EARLIER ( 'View Name'[Date] ),
    'View Name'[State Change Count] < EARLIER('View Name'[State Change Count - Last Proposed] ),
    'View Name'[State] <> "Proposed"
) + 0
```

Since the *State Restart Time in Days* is updated for each row of data, you can either create a trend to evaluate rework across specific sprints or examine rework for individual work items by using 'Is Current'.

Add *State Rework Time in Days*

Similar to *State Restart Time in Days*, the *State Rework Time in Days* looks for the first time a work item was in the Completed state category. After that time, each day a work item spends in a state other than Completed, counts as rework.

From the **Modeling** tab, choose **New Column** and then replace the default text with the following code and click the ✓ checkmark.

DAX Copiar

```
State Rework Time in Days =
IF (
    ISBLANK ( 'View Name'[State Change Count - First Completed] ),
    0,
    CALCULATE (
        SUM ( 'View Name'[Date Diff in Days] ),
        ALLEXCEPT ( 'View Name', 'View Name'[Work Item Id] ),
        'View Name'[Date] <= EARLIER ( 'View Name'[Date] ),
        'View Name'[State Change Count] > EARLIER ( 'View Name'[State Change Count - First Com-
pleted] ),
        'View Name'[State] IN {"Completed", "Closed", "Cut" } = FALSE()
    ) + 0
)
```

ⓘ **Observação**

You may need to revise the above definition based on the workflow states used by your project. For example, if your project uses 'Done' in place of 'Closed'.

DAX functions

Additional information is provided in this section for the DAX functions used to created the calculated columns and measure added in this article.

- `CALCULATE` : This function is the basis for nearly all examples. The basic structure is an expression followed by a series of filters which are applied to the expression.
- `COUNTROWS` : This function, `COUNTROWS ( 'View Name' )` , simply counts the number of rows which remain after the filters are applied.
- `LASTDATE` : We apply the `LASTDATE` filter to an expression, for example `LASTDATE ( 'View Name'[Date] )` , to find the newest date across all rows in the table and eliminate the rows that do not share the same date. With the snapshot table generated by an Analytics view this filter effectively picks the last day of

the selected period.

- `MAX` : Returns the largest numeric value in a column, or between two scalar expressions. We apply `MAX ( 'View Name'[Date] )` , to determine the most recent date after all filters have been applied.
- `ALLEXCEPT` : Removes all context filters in the table except filters that have been applied to the specified columns. Essentially, `ALLEXCEPT ('View Name'', 'View Name'[Work Item Id])` reduces the rows in the table down to only those that share the same work item ID as the current row.
- `EARLIER` : Returns the current value of the specified column in an outer evaluation pass of the mentioned column. For example, `'View Name'[Date] < EARLIER ( 'View Name'[Date] )` , further reduces the data set to only those rows that occurred before the date for the current row which is referenced by using the `EARLIER` function. `EARLIER` does not refer to previous dates, it specifically defines the row context of the calculated column
- `ISBLANK` : Checks whether a value is blank, and returns TRUE or FALSE. `ISBLANK` evaluates the current row to determine if *Date Previous* has a value. If it does not, the If statement sets *Date Diff in Days* to 1.
- `DATEDIFF` : Returns the count of interval boundaries crossed between two dates. `DATEDIFF` subtracts *Date Previous* from *Date* to determine the number of days between them.
- `LOOKUPVALUE` : Returns the value in *result_columnName* for the row that meets all criteria specified by *search_columnName* and *search_value*.

Related articles

- Power BI integration overview
- Create Analytics views
- Get started with Power BI Desktop
- Dataset design for the Power BI Connector
- Workflow states and state categories
- Data model for the Analytics service