

Lab 6. Task overdue

Author: Serge Luca aka "Doctor Flow"

Updated by: Dattatray-Patil

Learning objective: Controls, manipulation Excel, conditions, Date & time, expressions

Duration: 20 minutes

Scenario: We have an Excel document with a set of tasks, and some of these tasks are overdue. You will create a Flow that will find all overdue tasks and send a report of these tasks.

Tasks:

1. Create a new Excel file that looks like this in your OneDrive for Business (use the same columns). Before adding data in the Deadline column, ensure it is in **Text** format (otherwise, your flows will crash!). Create a table, and replace the existing e-mails with your e-mail.

| Task | Status | in charge | Deadline |
|------------------|-------------|-----------------------------------|----------|
| Feed the cat | Started | User1@dynamicjuly.onmicrosoft.com | 1/1/2021 |
| Call jon | Not Stared | User1@dynamicjuly.onmicrosoft.com | 2/4/2021 |
| Patch sql server | Started | User1@dynamicjuly.onmicrosoft.com | 7/6/2021 |
| Call mum | Started | User1@dynamicjuly.onmicrosoft.com | 6/7/2021 |
| Buy fruits | Started | User1@dynamicjuly.onmicrosoft.com | 8/8/2021 |
| Call Kent | Not Started | User1@dynamicjuly.onmicrosoft.com | 8/8/2021 |
| Buy a Porsche | Done | User1@dynamicjuly.onmicrosoft.com | 1/9/2021 |

| Task | Status | in charge | Deadline |
|------------------|-------------|-----------------------------------|----------|
| Feed the cat | Started | User1@dynamicjuly.onmicrosoft.com | 1/1/2021 |
| Call jon | Not Stared | User1@dynamicjuly.onmicrosoft.com | 2/4/2021 |
| Patch sql server | Started | User1@dynamicjuly.onmicrosoft.com | 7/6/2021 |
| Call mum | Started | User1@dynamicjuly.onmicrosoft.com | 6/7/2021 |
| Buy fruits | Started | User1@dynamicjuly.onmicrosoft.com | 8/8/2021 |
| Call Kent | Not Started | User1@dynamicjuly.onmicrosoft.com | 8/8/2021 |
| Buy a Porsche | Done | User1@dynamicjuly.onmicrosoft.com | 1/9/2021 |

2. If you already have existing data, change the format to Text :

| | A | B | C | D | E | F |
|----|------------------|-------------|--------------------------------------|------------|---|---|
| 1 | | | | | | |
| 2 | tasks | Status | in charge | Deadline | | |
| 3 | Feed the cat | Started | sergeluca@doctorflow.onmicrosoft.com | 1/1/2019 | | |
| 4 | Call Jon | Not Started | sergeluca@doctorflow.onmicrosoft.com | 10/10/2019 | | |
| 5 | patch SQL Server | Started | sergeluca@doctorflow.onmicrosoft.com | 7/6/2019 | | |
| 6 | Call Mum | Started | sergeluca@doctorflow.onmicrosoft.com | 6/7/2019 | | |
| 7 | Buy fruits | Done | sergeluca@doctorflow.onmicrosoft.com | 8/8/2019 | | |
| 8 | Call kent | Not started | sergeluca@doctorflow.onmicrosoft.com | 8/8/2019 | | |
| 9 | Buy a Porsche | Done | sergeluca@doctorflow.onmicrosoft.com | 1/9/2019 | | |
| 10 | | | | | | |
| 11 | | | | | | |

3. Name it **Tasks.xlsx**
4. Change the value of the **in-charge column** to your e-mail address and adjust some deadline value.
5. We want to write a Flow that will loop through all tasks, and that will check if the task is overdue. To do so, **create a new Instant cloud flow** and use a **Manually trigger a flow** as a trigger. Name it **my Overdue Tasks**.
6. Define a variable of type Array y adding an **Initalize variable** action as illustrated hereafter:

Manually trigger a flow

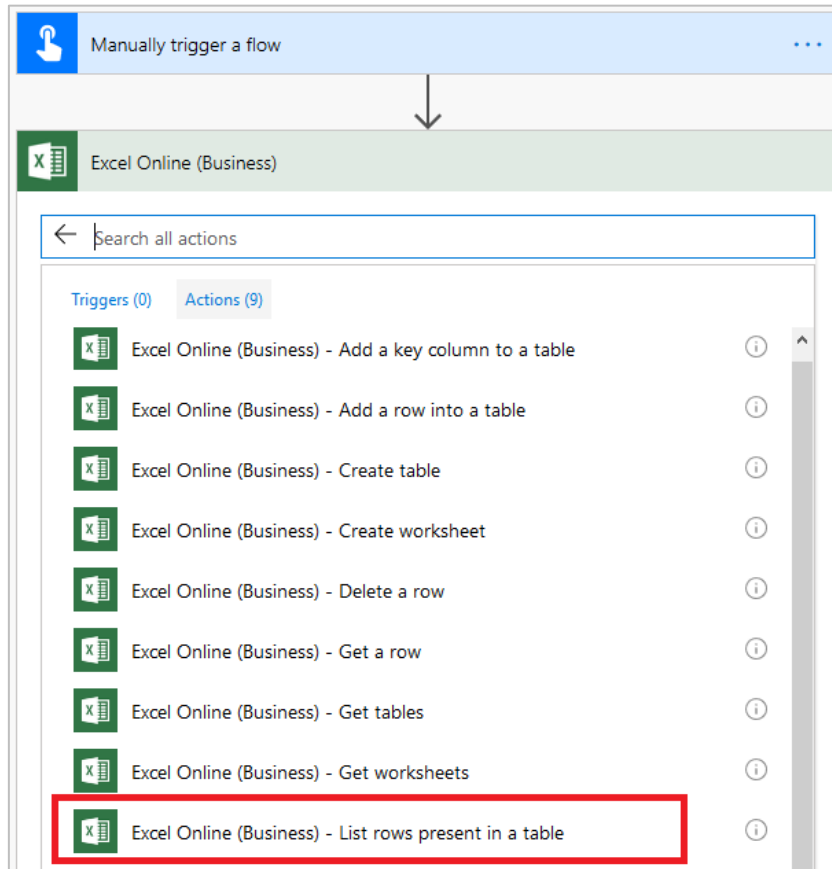
Initialize variable - Tasks Overdue

* Name: Tasks overdue

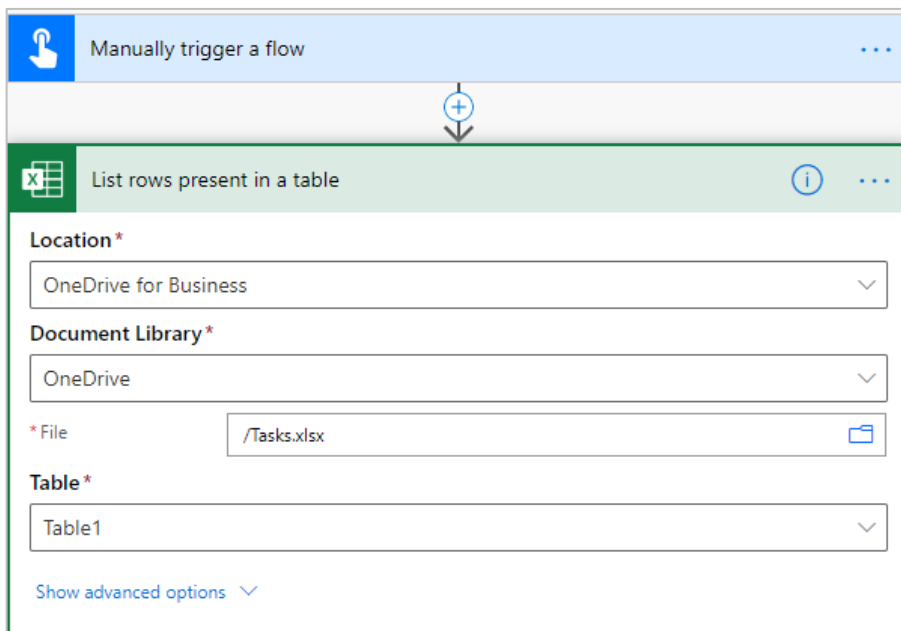
* Type: Array

Value: Enter initial value

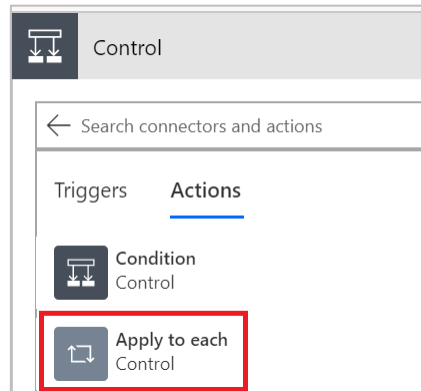
7. The Flow needs to connect to the **Tasks.xlsx** file, so add an **Excel Online (Business) – List rows present in a table** action (Not OneDrive !!!) as illustrated below:



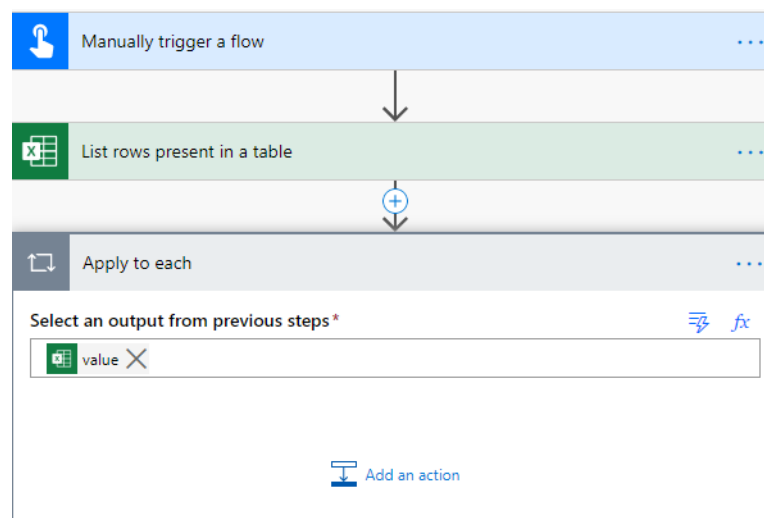
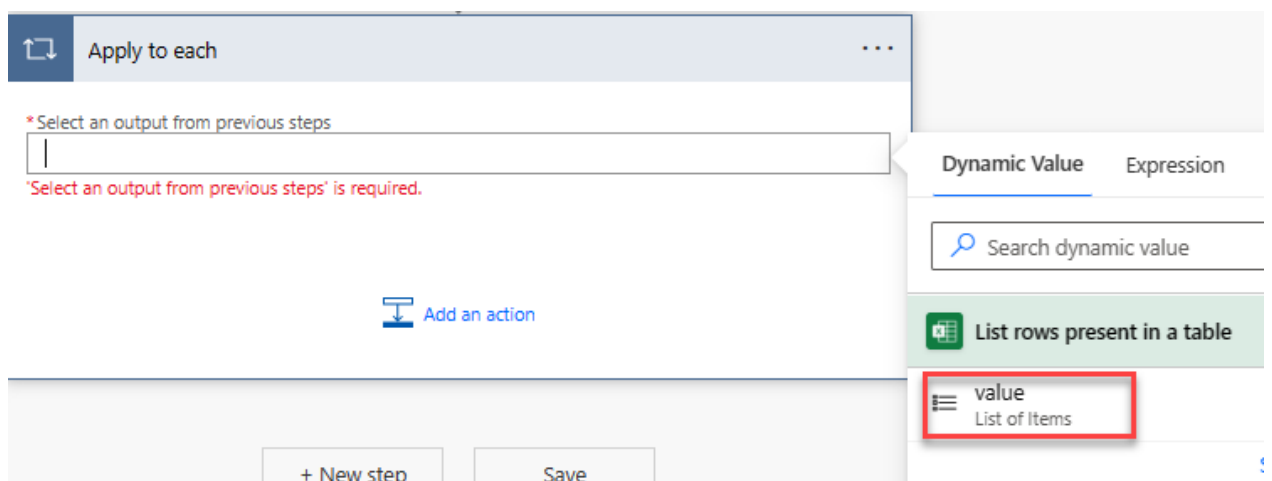
8. Set the action properties, as illustrated in the following picture:



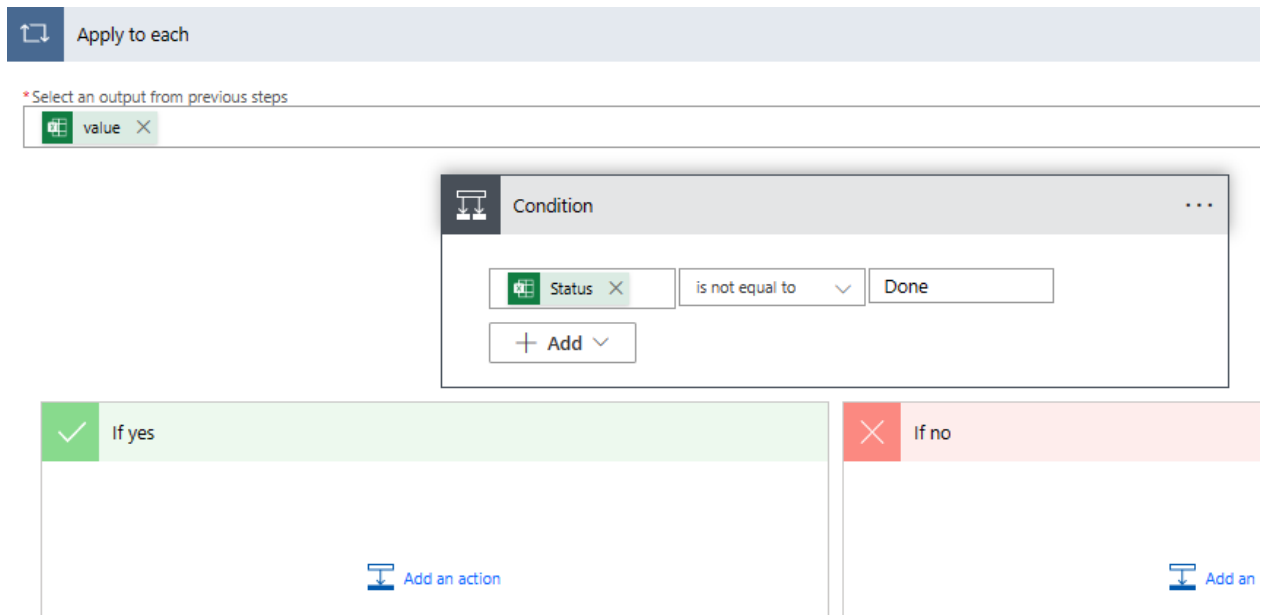
9. Let's loop through all tasks, so we need to add an **Apply to each**:



10. Select the add a **Dynamic value** in the **Apply to each** action:



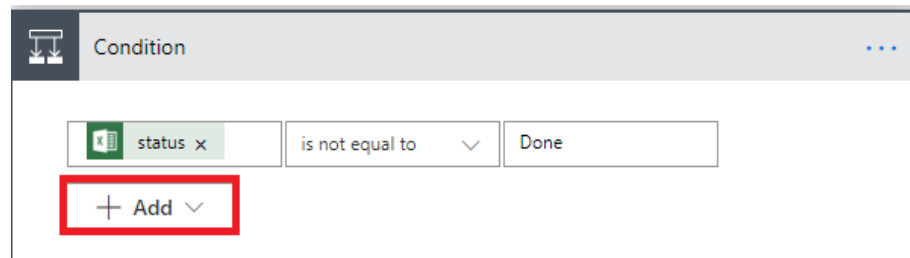
11. Add a **Condition** to filter the task where **Status** is not equal to **Done**:



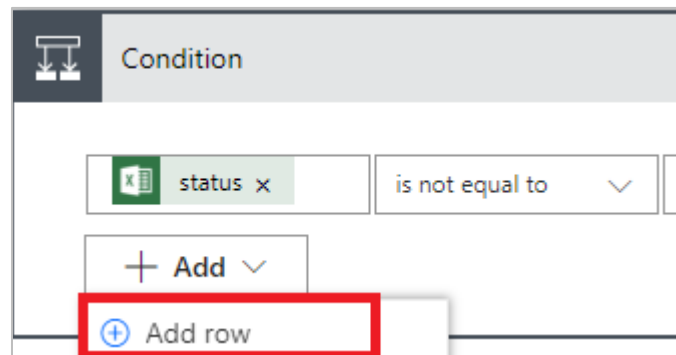
Make sure the operator used is **is not equal**.

12. Add a new sub condition that will check if the task is overdue:

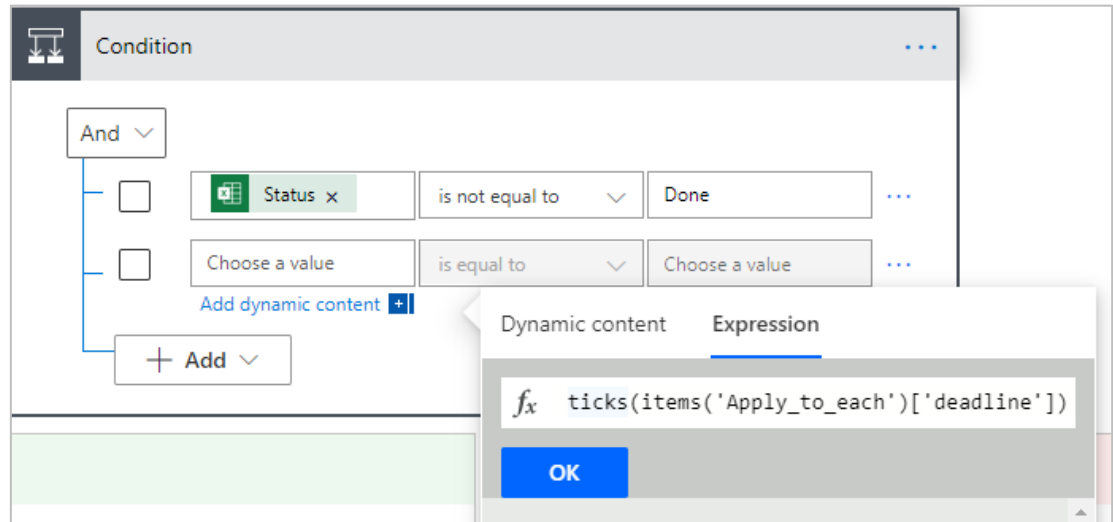
a) Click Add:



b) Select Add row:

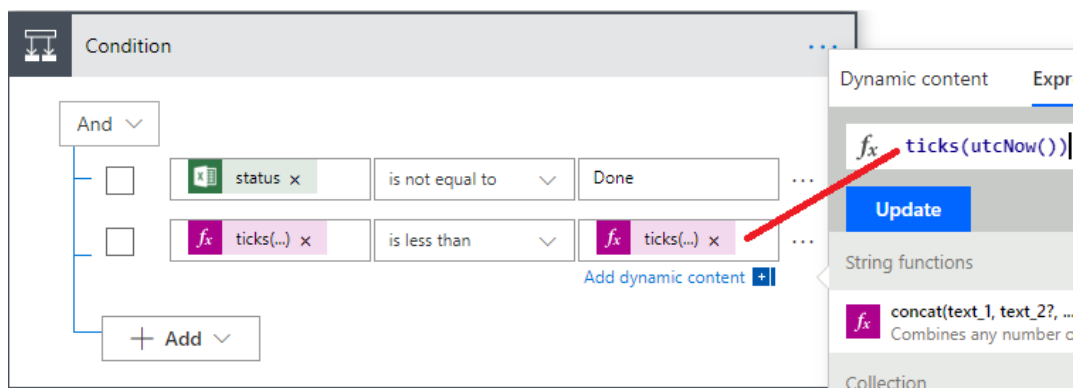


c) In "Choose a value", type the expression (**ticks()**), and In the parentheses, grab the dynamic value for Deadline and click Ok:



Since Deadline is a timestamp and has a string format, the **ticks** expression returns the number of ticks (100 nanosecond intervals) since 1st January 1601. By using ticks, we can compare two different timestamp values.

- d) Select the operator is less than
- e) Type the following expression (and click Ok):



13. In the **If yes branch**, add a new **condition** where we will check if the due date is overdue:

14. Rename the condition to **Check if overdue** to provide your Flow with more clarity:

Check if overdue

And

- ☐ Status is equal to Done
- ☐ ticks(...) is less than utcnow()

+ Add

15. In the If yes branch, add an **Append to array variable** action with the following JSON text (don't forget the commas, where it makes sense):

✓ If yes

Append to array variable

* Name: Tasks overdue

* Value:

```
{
  "Task":,
  "Deadline:",
  "Status:",
  "In charge:"
}
```

16. Add the dynamic value of **tasks**, **Deadline**, **Status** and **In charge** into this action:

✓ If yes

{x} Append to array variable

* Name Tasks overdue

* Value

```

{
  "Task": tasks ,
  "Deadline": Deadline ,
  "Status": Status ,
  "In charge": in charge
}

```

17. After the **Apply to each** add a Create HTML table action where you will generate a report:

↩ Apply to each

{ Create HTML table

* From {x} Tasks overdue

* Columns Custom

| Header | Value |
|-----------|-----------------|
| Task | {x} item(...) X |
| Deadline | {x} item(...) X |
| In charge | {x} item(...) X |
| Status | {x} item(...) X |

18. Here is how the Task value has been implemented:

The screenshot displays a workflow editor with the following steps:

- Initialize variable - Tasks Overdue** (purple box)
- List rows present in a table** (green box)
- Apply to each** (blue box)
- Create HTML table** (purple box)

The **Create HTML table** action is configured with:

- From:** Tasks overdue
- Columns:** Custom

The table structure is as follows:

| Header | Value |
|-----------|------------------------|
| Task | <code>item(...)</code> |
| Deadline | <code>item(...)</code> |
| In charge | <code>item(...)</code> |
| Status | <code>item(...)</code> |

The right-hand pane shows the **Dynamic Value** tab with the expression `item()?['Task']`. A red arrow points from the `item(...)` expression in the first row of the table to this expression in the pane.

Run the flow and check the output of the Create HTML table action:

Create HTML table

0s

INPUTS

Show raw inputs >

Format

Html

From

```
{
  "Task": "Feed the cat",
  "Deadline": "01/01/2019",
  "Status": "Started",
  "In charge": "Started"
},
```

OUTPUTS

Show raw outputs >

Body

| Task | Deadline | In charge | Status |
|------------------|------------|-------------|-------------|
| Feed the cat | 01/01/2019 | Started | Started |
| Call Jon | 01/01/2020 | Not Started | Not Started |
| patch SQL Server | 01/01/2021 | Started | Started |
| Call Mum | 01/01/2022 | Started | Started |

19. We can drastically simplify this flow : instead of filtering the Tasks with the Status Done in a condition, you can also keep it more straightforward by using an OData Filter (click on Show advanced options) in the List rows present in a table action:

List rows present in a table

Location *

OneDrive for Business

Document Library *

OneDrive

* File

/Tasks.xlsx

Table *

Table1

Filter Query

Status ne 'Done'

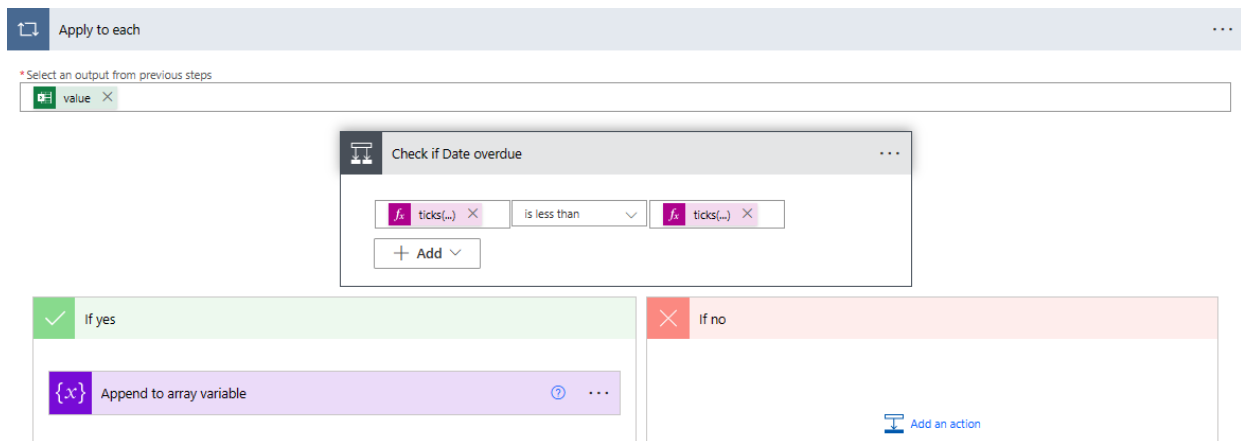
Order By

Using OData Filter (Filter query) is often considered as a good practice because the data are filtered on the server-side, and that can make your Flow running faster by using fewer resources.

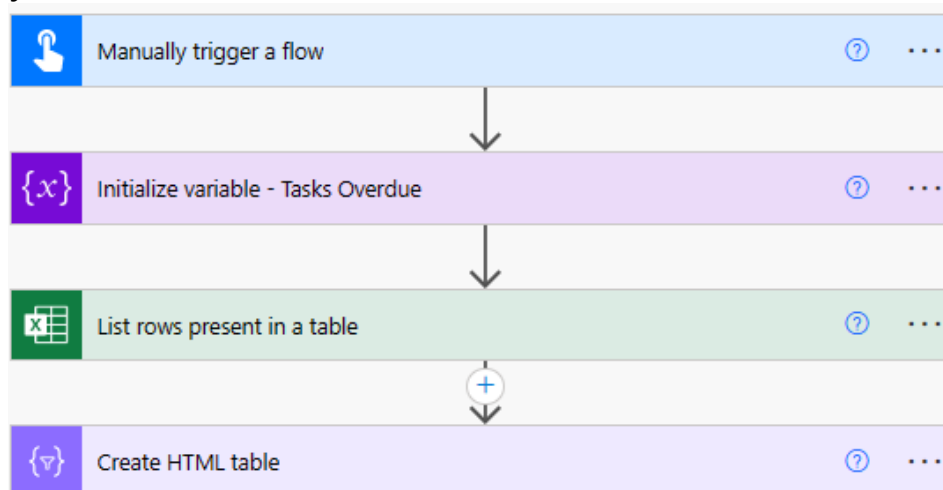
Note: oData expressions are case sensitive!

How can this simplify how flow ?

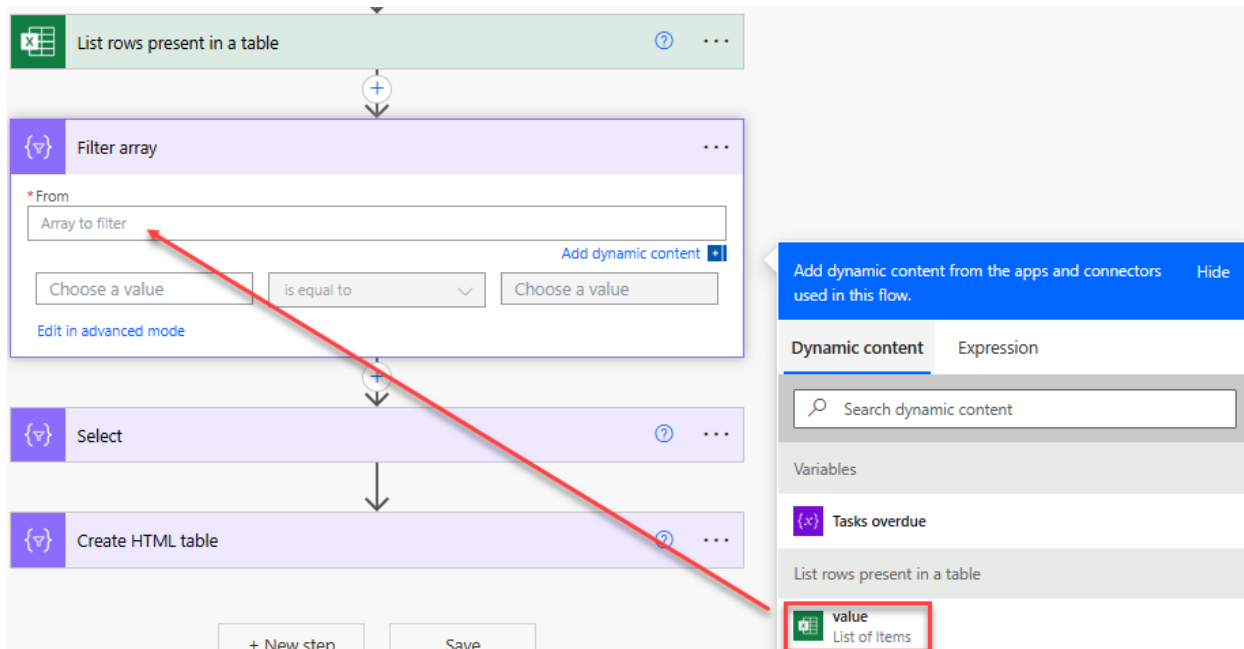
We can remove the **is not equal to Done** condition:



20. But the oData filter does not allow us to compare two dates. Nevertheless, we can **remove** the **Apply to Each**:

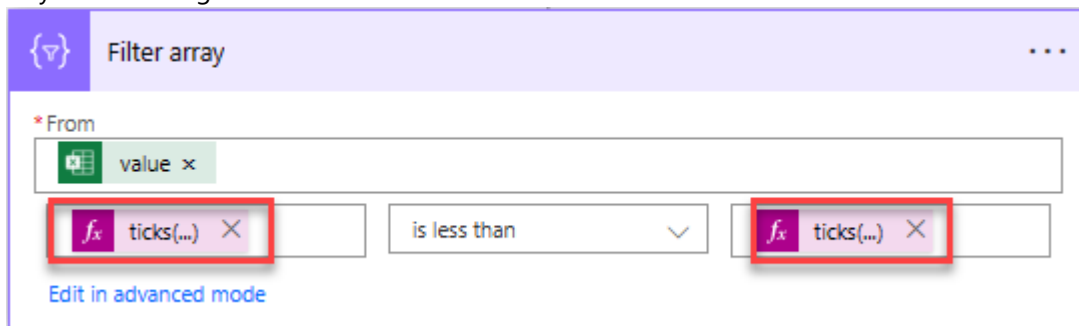


But we still need to feed the Create HTML table action by adding a Filter Array action that we will **connect** to the **List rows present in a table** action:

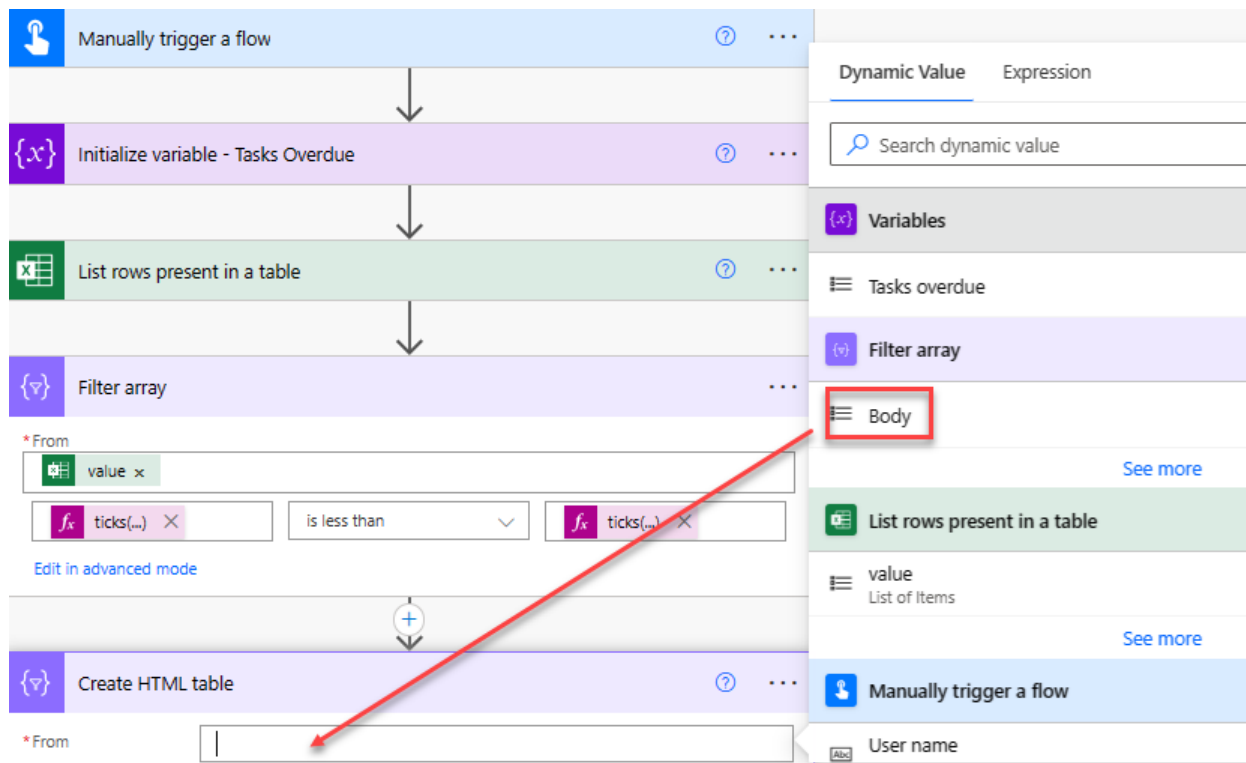


In some cases, filtering data in the Flow itself cannot be avoided. In this case, you can use the Filter Array action instead of adding conditions. The ticks() expression defined before can be reused in the Filter array:

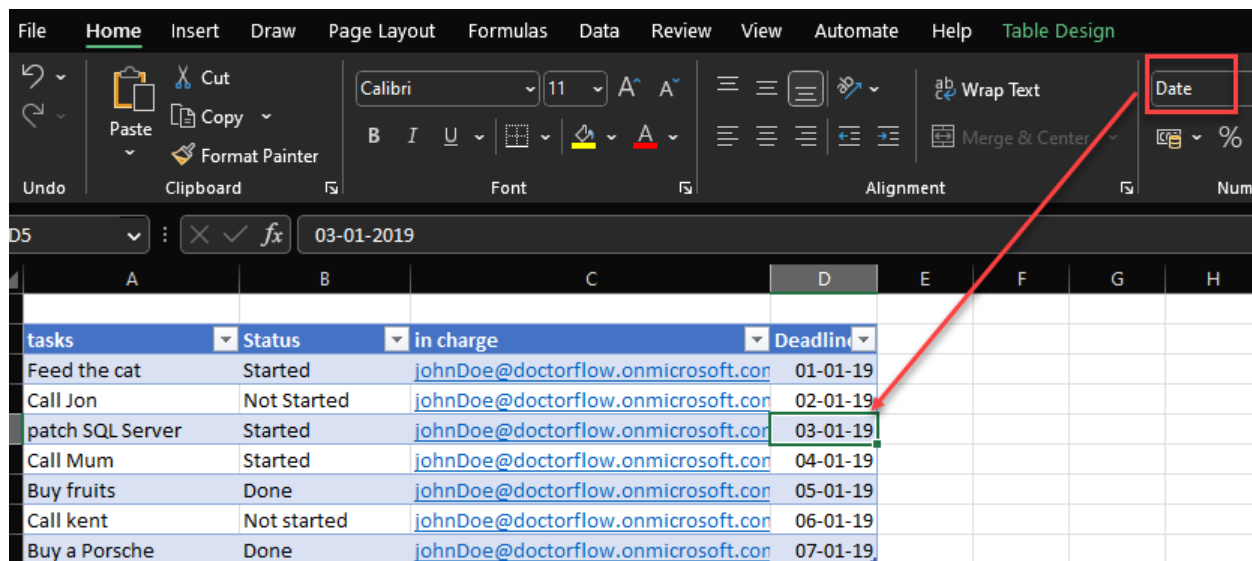
21. Try the following code:



22. We still need to **connect** the **Create HTML table** action and the **Filter array**, by using its **Body** property:



23. You will notice that the Deadline format in the Excel document is in a text format, but if we switch this format to a Date format (and re-enter the dates), our flow will crash:



Indeed, if we switch the Deadline column type to **General** and if re-write (or just update) the data our flow run will generate an error:

Error Details

The execution of template action 'Filter_array' failed: The evaluation of 'query' action 'where' expression '@less(ticks(item()['Deadline']), ticks(utcNow()))' failed: 'In function 'ticks', the value provided for date time string '43466' was not valid. The datetime string must match ISO 8601 format.'

24. Deadline field is now a number that give you the number of days between 1899-12-30 and your specific date. The ticks() function expects a real date, so it fails here. It should be updated with:

```
ticks(addDays('1899-12-30',int(items('Apply_to_each')['deadline'])))
```

25. You can test this using the Excel file **Tasks with Date format.xmls** in the provided in the resources.zip file.

We need your feedback

Do you want to report an issue or suggest something? We need your feedback:

<https://github.com/Power-Automate-in-a-day/Training-by-the-community/issues>