Power Automate and Power BI

Module 2

Every month, we need to keep our records up to date. However, we are quite forgetful. That is why we would like to receive a monthly notification as a reminder to do our administration.

1) Create a flow based on a monthly repeat.
2) Send a (mobile) notification with the text: “Time for the administration!”
3) Test the flow by downloading the “Power Automate” app and starting it there.
4) Also try starting the flow once using the “Test” button. Can this be done “automatically”? Why do you think so or not?

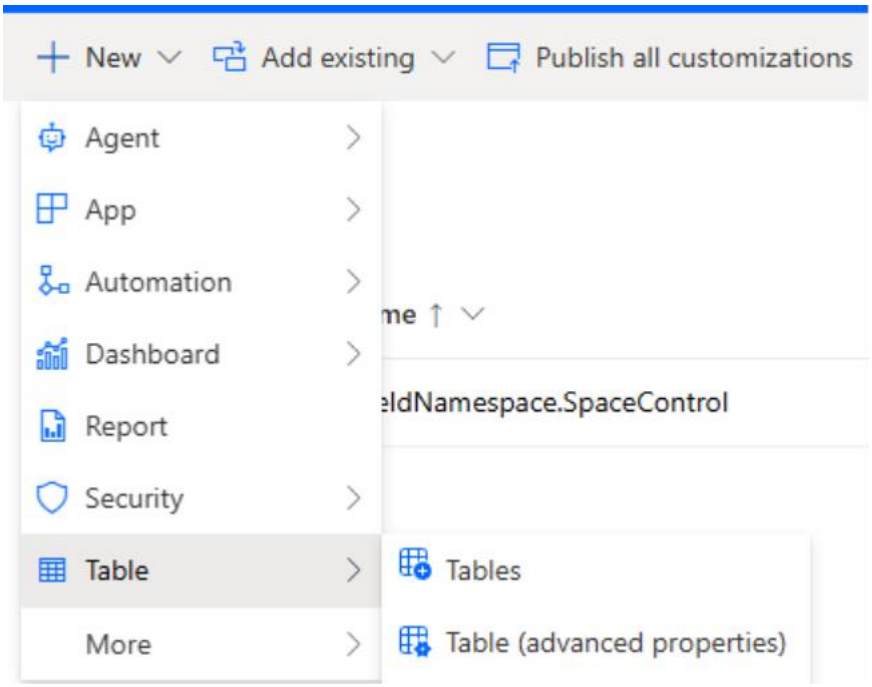
Module 3

We want to use solutions for our sales database, in which we want to create a Products table. We want to be able to transport these.

(If your license allows it): Create a new development (developer) environment, containing a Dataverse. If this is not possible in the practice environment, we will use the existing environment.

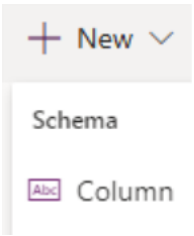
1) Go to the solutions screen, refresh (F5), and ensure that your new environment is selected. Create a new solution and a Publisher that suits your needs. Only the prefix is important for recognizing the objects you have created.

2) Create a table in your Dataverse. Do this via the ‘solutions’ screen (select ‘Solutions’ in the left-hand menu). Select your solution there, then select ‘+New’ > ‘Table (Advanced Properties)’ from the menu:



Name the table ‘Product’ (plural = Products). Click on ‘Save’.

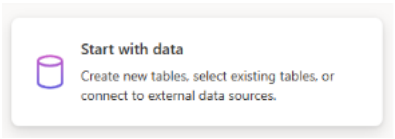
3) Now create a new column called ‘Price’ in the table you just created. This can be a useful type such as ‘Currency’.



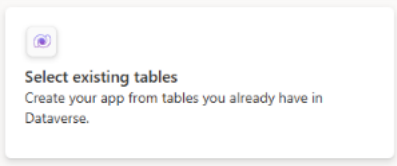
4) Go to: https://make.powerapps.com. This is the home page where we can build apps. We are going to try that now.

   Make sure you are working in the correct environment again and that you are on the ‘+Create’ page in the left-hand menu.

   Select ‘Start with data’:

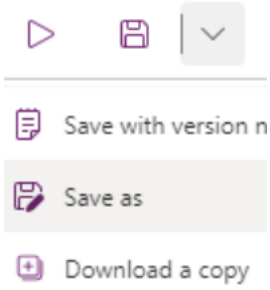


and then ‘Select existing tables’:



Select your ‘Product’ table and click on ‘Create App’.

5) Once the app has been created, go to the ‘File’ menu at the top left and select ‘Save’ or ‘Save as’. (The app is now called ‘App’:



6) Go back to the https://make.powerapps.com site and add the table and App you just created to your Solution by selecting ‘Solution’ in the left-hand menu, selecting your solution, and then selecting ‘Add existing’ and ‘Canvas app’ in the top menu. Then select the app you just created in the tab. From Dataverse Click ‘Add’.

7) Select ‘Publish all customizations’. Why wouldn't you want to use this button for exporting?

8) Check whether your solution is OK using the solution checker. This is optional! It can take a long time! Once it is ready, you will receive a message at your admin email address. Allow 15 minutes for a small solution.

   The solution checker checks the following topics, among others:
   - Dataverse plugins (extensions using code)
   - Dataverse custom workflows
   - Dataverse web resources
   - Dataverse configurations (also regarding access labels, etc. in apps

   There may only be ‘medium’ or ‘low’ severity notifications regarding the Canvas app. You can ignore these for now.

9) Now export your solution as an unmanaged solution. You can import it back into another (dev/test) environment for practical use.

10) Create a new patch for the current solution. To do this, from your dev environment:
    - Select your current solution by clicking on the circle in front of it.
    - In the menu, select the option 'Clone' > 'Clone a patch'.
    - If necessary, choose a clear name and leave the version number as it is (this will change automatically).

11) Add your existing 'Products' table to this Patch. Now modify this table by adding a column with a distinctive name such as #special_column.

12) Merge this patch with the original solution by:
    - Selecting your original solution using the 'dot' in front of it
    - Selecting the option 'Clone'>'Clone solution' in the menu
    - Leave the version number as it is

13) Now check your updated solution to see if the new column has been added to the table.

Module 4

Exercises A

We have an Excel sheet with planned events. When the flow is started, all these outings should appear as events in our Outlook calendar on the correct day.

1) Create an Excel sheet in Excel Online (in OneDrive for Business location). If the OneDrive webpage is not easily accessible because we are working with a dev/trial environment, you can also save the Excel file locally and then upload it.

2) Create at least four columns:
    a. Title
    b. Description
    c. Start date
    d. End date

3) Fill in the table with at least three rows. Ensure that the start date and end date columns are structured according to this format (and therefore also contain a time): 2022-06-23T12:00:00. Do not create a date column for this in Excel.

4) Select your entire table (all cells in the table) and select ‘Table’ on the ‘Insert’ tab and indicate that the table contains column headers.

   Close Excel again.

5) Create a flow based on an Instant trigger (‘manually trigger a flow’).

6) Then use an action that retrieves all rows from the relevant Excel sheet (‘list rows present in a table’ from the ‘Excel Online (Business)’ connector).

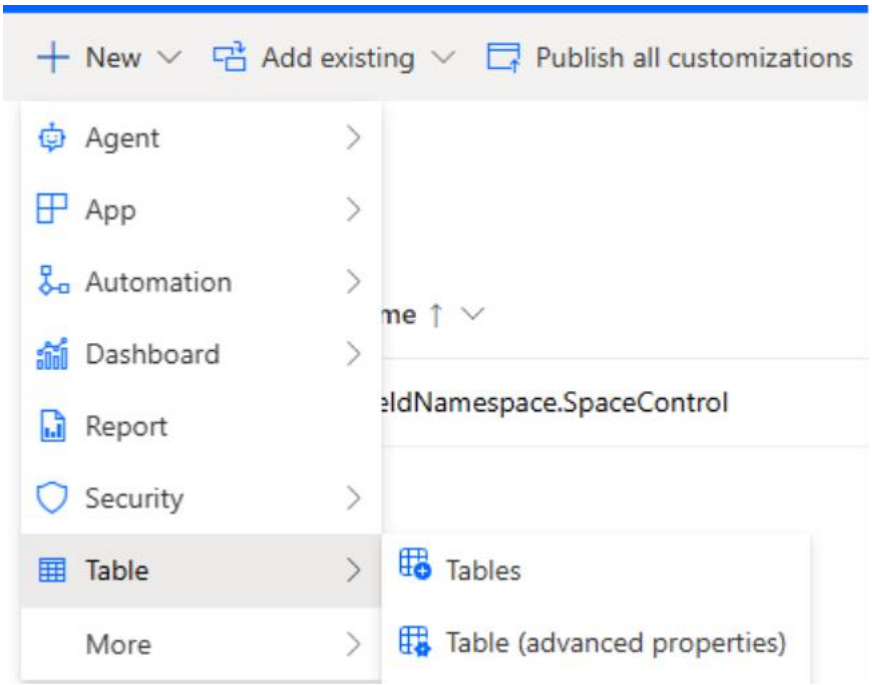
7) By building a flow, create a new event in your Outlook calendar for each item in the Excel sheet with the title of your outing from the Excel sheet as the subject. Do the same for the start and end times. There is no need to check whether the event already exists. You can initially choose the action for creating a new Event.

   Please note! An ‘Apply to each / for each’ step will then be created automatically. You can add the data in the ‘Apply to each’ step to create the Calendar item.

Exercise B

We want to know which customers have not ordered anything for more than a year. We first need to register these customers in a Dataverse table. For the overview, we have decided to register both the customers and the last order date in the same table.

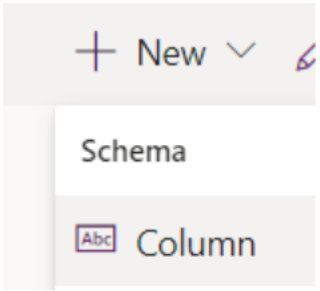
1) Navigate to your solution to create a new table to store customer data:



2) Name the table ‘Customer’ and, if necessary, change the plural name to ‘Customers’.

3) You can change the primary column (tab two) to ‘customer code’. Click <Save>.

4) Add a column named ‘First name’ of the type ‘Single line of text’ with format ‘Text’ and make it ‘required’:

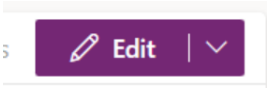


5) Add a column called ‘Last name’, also of the type ‘Text’.

6) Add a column called ‘Email’ of the type ‘Single line of text’ with the type ‘Email’ and make it ‘required’.

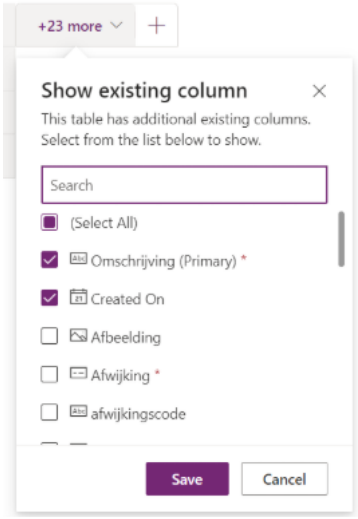
7) Add a column called ‘Last order’ with the data type ‘Date and time’ and the format ‘Date only’.

8) Now select ‘Edit’:



You are now in a screen for adding and editing rows.

9) Make sure you see all the columns you just created in the table by adding them here:



10) Now add at least 3 rows to the table, with at least one of the rows having a date more than a year ago. A row is automatically saved as soon as you navigate to another (or new) row.

11) Now create an instant cloud flow in Power Automate. As your first action, retrieve all rows from the table you just created.

12) Now go through the table and for each row where the date in the ‘Last ordered’ column is greater than: now minus 365 days, send an email to that person (in real life, of course).

Tip: use a condition with the utcNow() function to retrieve ‘now’.
Tip 2: adding days is not done with plus (+) or minus (-), but there is a function for that too. Use Google or (e.g.) ChatGPT to find out what it is.

13) Try the previous assignment again, but this time use an OData query. Search Google (or ask ChatGPT) to find out what this is; it will be discussed in a later chapter.

Could this also be done with a ‘Filter Array’ data action? What are the advantages or disadvantages?

Module 5

Exercise A

The sales department too often posts quotes in Sharepoint that do not meet the requirements set by managers in terms of amounts. It would also be helpful for managers to receive a notification when a quote has been drawn up.

1) Create a Sharepoint list in which we will keep track of the quotations. For now, these only need to be names, quantities, and prices. Navigate to Sharepoint and select a Site (or create one if you don't have one yet). See the following exercise for the names of the columns. Are you unfamiliar with Sharepoint? Ask the trainer for an explanation.

2) Make sure you have at least the following columns:
   - Customer name (single line of text)
   - Product quantity (number, 0 decimal places)
   - Quotation price (number, € symbol, automatic decimal places)

3) Create a flow that starts as soon as a new row is added to the SharePoint list.

4) Add an approval requesting approval from the manager of the user who added the item to SharePoint. For now, choose your own environment email address. You can choose to only send an approval if the quote price column is above 2000.

5) If approval is not granted, an email must be sent to the person who added the row to the SharePoint list. For bonus points, you can try to remove the added item.

6) Instead of separate columns in Sharepoint, create a file upload for a PDF file. Create a few sample invoices in Word and save them as PDF files to OneDrive for Business. Use a Power Platform AI Builder model to read the file using OCR when it is uploaded and send the approval if the amount exceeds 2000.

Exercise B

Working with functions is necessary in almost every flow. Conditions are also encountered in virtually every flow. A number of (relatively) short assignments have been created to practice with both.

For the assignments below, use ‘input’ from the instant trigger and, for example, a ‘compose’ step to perform the actions. You can send the result to yourself in a notification or view it in the ‘run history’.

1) Create a flow that takes an email address as input and extracts the domain name. Use the substring() and indexOf() functions to extract the domain from the email.

2) Create a flow that takes a string with a list of comma-separated names and converts it into an array. For example, use the join() function to create a new string with the names separated by semicolons. For example, the input “Anne, Qwin, Sophie” should be converted to “Anne; Qwin; Sophie”.

3) Create a flow that takes a date as input and determines whether it is a weekday or a weekend. Use the dayOfWeek() function to find the day of the week and compare it to the integer values for Saturday and Sunday.

4) Create a flow that takes a timestamp as input and formats it into a readable string with the format “Month Day, Year - Hours:Minutes”. Use the formatDateTime() function to achieve this.

5) Create a flow that takes a number as input and checks whether it is divisible by 3 and 5. Based on divisibility, we will fill in a text variable. If the number is divisible by 3 and 5, “Hurray” is set. If it is divisible by 3, ‘How’ is entered; if it is divisible by 5, “ra” is entered. In all other cases, “snif” is entered. For example, use the if() and/or equals() functions together with the mod() function.

6) Create a flow that takes a row of numbers as input and calculates the sum and average of the numbers. Use the length() function to find the number of elements and the add() function in a loop to calculate the sum.

7) Create a flow that takes a string as input and reverses it. For example, the input “Hypetrain” should be reversed to “niartepyH”. Use the substring function and a loop to reverse the string.

Module 6

We recently discovered the concept of error handling using scope try-catch blocks and would like to apply this in our existing and new flows.

1) Use a previous flow in which you executed an HTTP request. Set up a retry policy here. Haven't used an HTTP action yet? Then create a flow that retrieves the next SpaceX flight daily at 10:00 a.m. and sends it to you in a notification (this can be plain JSON text for now). Use this URL: https://api.spacexdata.com/v5/launches/latest

   Please note: if you want to extract something useful from this data, you will need a 'parseJSON' action.

   Note: the above API may not always provide consistent output, causing your parseJSON action to return an error. In that case, use a different API that your instructor can provide.

2) See which existing flows you can extend with error handling using run-after settings. If the flow is large, you can also use scope blocks to set up a try-catch-finally construction.

Module 8

1) Start Power BI desktop and create a new empty report.

2) Look up a dataset (preferably .xlsx file) you like from https://www.kaggle.com/datasets. Download the dataset to your local machine and connect to this data source from Power BI desktop.

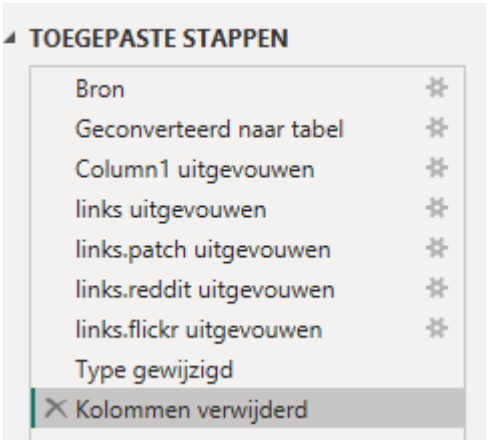
3) Close the current report, save it if you like, then create a new report.

4) Look up the SpaceX public JSON API, version 5. Now connect to this JSON data source from Power BI desktop. For example, use this URL: https://api.spacexdata.com/v5/launches/

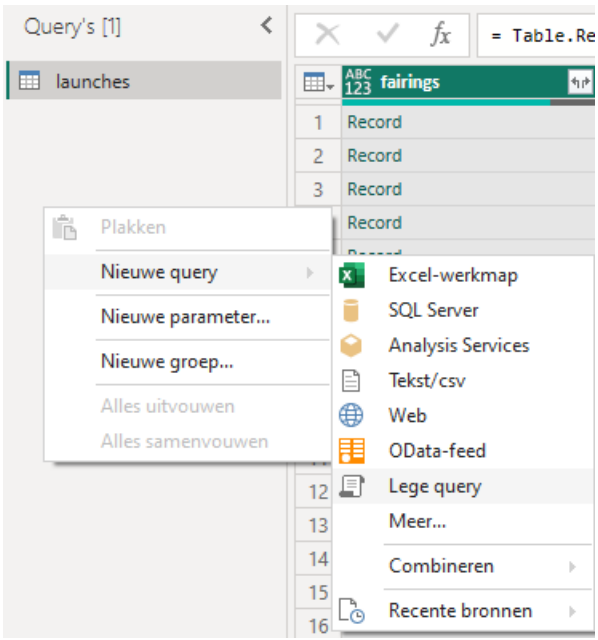
Module 9

1) Remove the columns called 'auto_update', 'tbd', 'launch_library_id' and 'id'.

2) Navigate through the querysettings applied steps and see which steps have been done that you didn't knowingly take:



3) Create a new query in PowerQuery where you filter the data and take the other query as basis:



As function in the query bar, use the following formula, where you replace 'Fill-in-Dataset-Name' with your dataset (called 'launches' in the above screenshot):

= Table.SelectRows(#"Fill-in-Dataset-Name", each [fairings] <> null)

Then press <Enter>. What did this formula do? Rename this Query to 'Final' in the 'Query' window on the left side of the screen.

4) Now apply your queries and close PowerQuery.

Module 10

1) Create a report from Power BI desktop, add at least three different kind of visuals that represent relevant data.

2) Now publish the report.

3) Create a new dashboard in the Power BI cloud service and add a few tiles from the report you just published.