



# Power Apps

## Advanced

# Power Apps Advanced

---

Copyright © 2021 Excel Consulting Solutions Pty Ltd. T/A Nexacu  
Published by Nexacu



Brisbane | Melbourne | Sydney | Adelaide | Perth | Canberra | Parramatta  
Australia

Phone: (+61) 1300 886 190

Web: [www.nexacu.com.au](http://www.nexacu.com.au)

Email: [info@nexacu.com.au](mailto:info@nexacu.com.au)

**Product Code: PA Level 3 v 2.0**

**Build: 1/08/2021**

---

## Trademark Acknowledgments

All terms mentioned in this manual that are known to be trademarks or service marks have been appropriately acknowledged or capitalised. Nexacu cannot attest to the accuracy of this information. Use of a term in this manual should not be regarded as affecting the validity of any trademark or service mark.

Screen Shots © 1983-2021 Microsoft. All rights reserved.

## Disclaimer

Every effort has been made to provide accurate and complete information. However, Nexacu assumes no responsibility for any direct, indirect, incidental, or consequential damages arising from the use of information in this document. Data and case study examples are intended to be fictional. Any resemblance to real persons or companies is coincidental.

## Copyright Notice

This publication is protected in accordance with the provisions of the Copyright Act. Apart from permissions expressed in the Copyright Act pertaining to copying for study, review, or research, no part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording, or otherwise without written permission from Nexacu.

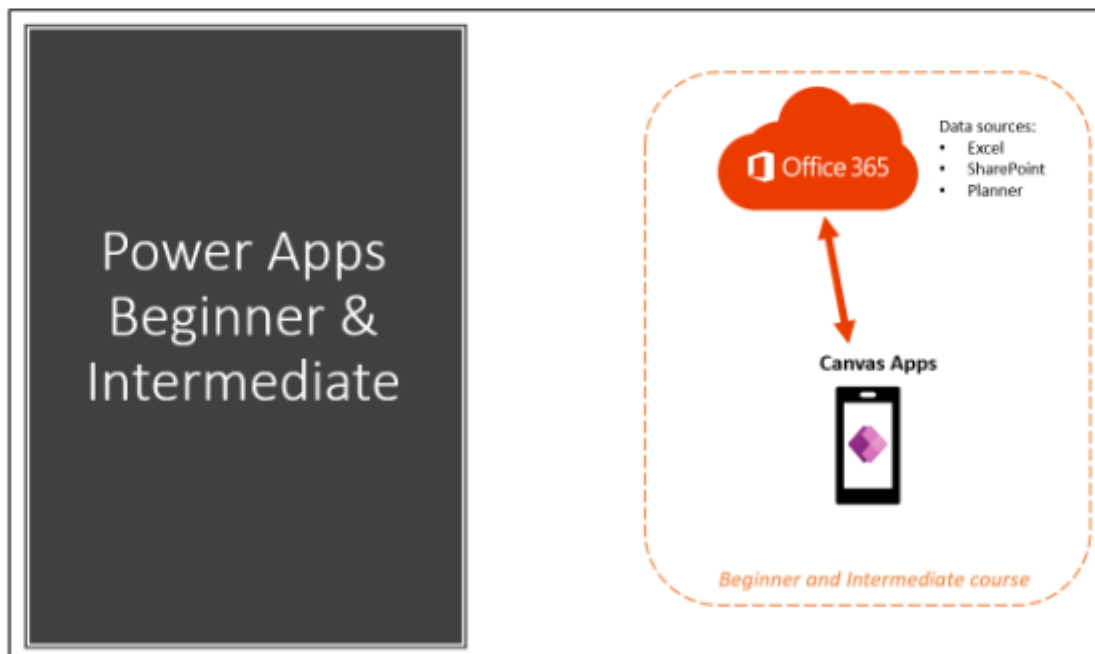
# Contents

|  |           |
|--|-----------|
| <b>1. THE POWER PLATFORM.....</b>              | <b>3</b>  |
| <b>2. LICENSING CONSIDERATIONS .....</b>       | <b>5</b>  |
| POWER APPS .....                               | 5         |
| PORTALS.....                                   | 6         |
| <b>3. BUSINESS CASE INTRO.....</b>             | <b>7</b>  |
| NEXACU GYM.....                                | 7         |
| PROCESS OF CREATING SOLUTION .....             | 7         |
| <b>4. DATAVERSE .....</b>                      | <b>8</b>  |
| SETTING UP DATAVERSE .....                     | 9         |
| TABLES.....                                    | 10        |
| COLUMNS .....                                  | 12        |
| BUSINESS RULES.....                            | 13        |
| MORE TABLES.....                               | 15        |
| ADDING DATA USING EXCEL.....                   | 17        |
| INTERACT WITH DATA IN THE BROWSER .....        | 18        |
| RELATIONSHIPS .....                            | 20        |
| MORE BUSINESS RULES .....                      | 21        |
| <b>5. POWER APPS CANVAS .....</b>              | <b>26</b> |
| IMPORTING AN APP .....                         | 26        |
| DISPLAYING THE APP VERSION WITHIN AN APP ..... | 27        |
| CONNECT TO THE DATAVERSE.....                  | 28        |
| CONNECT A GALLERY TO THE DATAVERSE.....        | 29        |
| CONNECT FORMS TO THE DATAVERSE.....            | 29        |
| SUBMITTING THE DATA.....                       | 32        |
| USING DATA TABLES .....                        | 33        |
| FILTERING DATA TABLES .....                    | 34        |
| <b>6. PREPARE FOR PORTAL EXERCISES.....</b>    | <b>36</b> |
| CREATING A PORTAL APP .....                    | 36        |
| <b>7. MODEL-DRIVEN APPS .....</b>              | <b>37</b> |
| CREATE A MODEL-DRIVEN APP.....                 | 37        |
| CREATE A NEW VIEW .....                        | 39        |
| FORMS .....                                    | 42        |
| <b>8. BUSINESS PROCESS FLOWS .....</b>         | <b>45</b> |
| USING POWER AUTOMATE .....                     | 45        |
| <b>9. BRINGING THE APP TOGETHER .....</b>      | <b>51</b> |
| PUBLISHING CUSTOMISATIONS.....                 | 51        |
| SHARING A MODEL-DRIVEN APP .....               | 51        |
| <b>10. POWER AUTOMATE .....</b>                | <b>55</b> |
| <b>11. PUTTING THE PROCESS TOGETHER.....</b>   | <b>57</b> |
| <b>12. PORTAL.....</b>                         | <b>58</b> |
| EXPLORING THE PORTAL DESIGNER .....            | 59        |
| EXPLORING THE PORTAL MANAGEMENT APP.....       | 61        |
| ADDING AND MOVING PORTAL PAGES .....           | 61        |
| DISPLAYING DATAVERSE DATA ON A PAGE.....       | 62        |
| RESTRICTING WHO CAN VIEW A PAGE .....          | 64        |

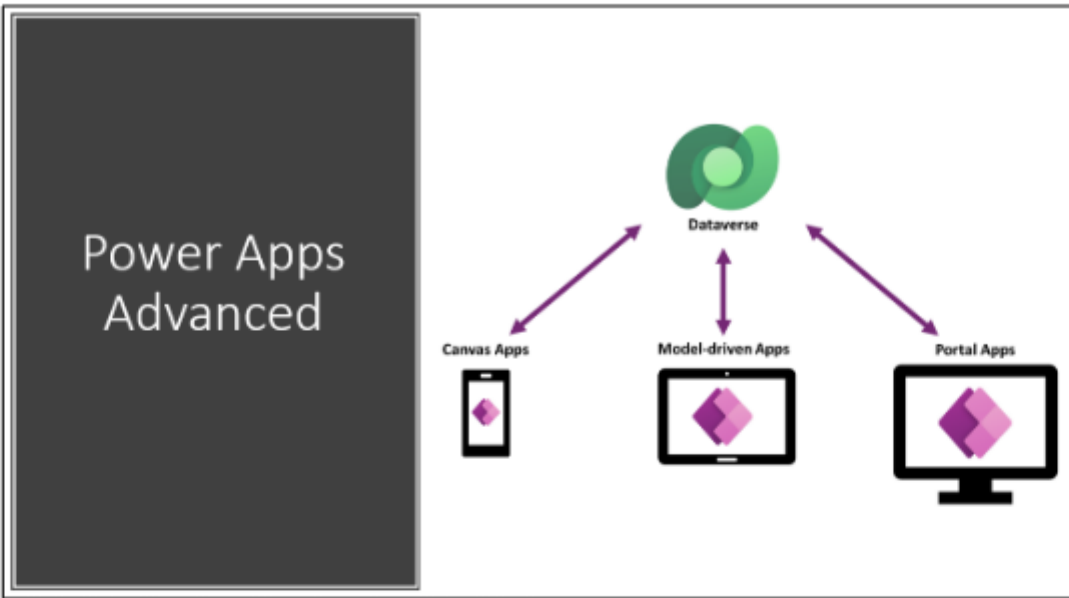
# 1. The Power Platform



The Power Platform provides a capability for organisations to act, automate, and analyse their data. Power Apps allows business users to quickly create applications that collect, store and manipulate business data.



In the previous Power Apps courses, we explored data outside of the Power Platform using Data connectors (SharePoint lists, Planner, etc).



Throughout the advanced course we are going to use the Dataverse as our data source. The Dataverse is the Power Platform’s own data source. Dataverse was previously known as the Common Data Service or CDS.

In the beginner and intermediate course, we explored and created canvas apps. During this course we will learn to use model-driven apps and portal apps.

| Client               | Connectors | Common Data Service |
|----------------------|------------|---------------------|
| Power Apps (Canvas)  | Yes        | Yes                 |
| Power Apps (Model)   | No         | Yes                 |
| Power Portals        | No         | Yes                 |
| Flows                | Yes        | Yes                 |
| AI Builder           | No         | Yes                 |
| Power BI             | Yes        | Yes                 |
| Power Virtual Agents | Yes        | Yes                 |

Figure 1 - Some 'clients' only work with the Dataverse

## 2. Licensing considerations

### Power Apps

**Power Apps pricing**

Australian Dollar (\$)

**Run single apps**

Run applications around a single business scenario for an individual employee.

**\$13.70**  
per user/app/month<sup>1,2</sup>

[Buy now >](#)

**Run unlimited apps**

Run unlimited applications for multiple business scenarios.

**\$54.90**  
per user/month<sup>1</sup>

[Buy now >](#) [Try free >](#)

| Type  | Cost                            | Description  |
|---|---------------------------------|--|
| <a href="#">Developer Plan</a>              | Free                            | Gives a user their own personal full featured sandbox environment including a Dataverse database. You can do everything including sharing for test and evaluation.                                     |
| <b>Power Apps for Microsoft 365</b>         | Included in most M365 licenses. | Enable users to extend and customize the Office experience with Power Apps and Power Automate. Users can create applications and flows based on Microsoft 365 data. Limited to non-premium connectors. |
| <b>Power Apps per app plan<sup>1</sup></b>  | US\$5/user/month                | Allow individual users to run <i>one</i> application for a specific business scenario based on the full capabilities of Power Apps including Dataverse and all connectors.                             |
| <b>Power Apps per user plan<sup>1</sup></b> | US\$10/user/month               | Equip users to run <i>unlimited</i> applications based on the full capabilities of Power Apps including Dataverse and all connectors.  |

<sup>1</sup> New pricing and entitlements for Power Apps licensing is shown above. Effective from 1 October 2021. Previously Power Apps per app allowed for 2 apps and a portal.



## Portals

For users outside the organisation there are additional costs for accessing the Portal App. This depends on if you are an authenticated user (users can login through Azure AD, LinkedIn, etc) or an anonymous user.

### Portals

Enable external users to access custom portals.

Login capacity  
(Authenticated users)

Starting at

**\$274.60**

per month for 100 daily login sessions\*

[Buy now >](#)

Page view capacity  
(Unauthenticated users)

**\$137.30**

per month for 100,000 page views

[Buy now >](#)

[Learn more >](#)

### Usage of a Power Apps Portals instance

Each end user that accesses a Power Apps Portals instance needs to be licensed appropriately. The table below outlines the end user types.

| End user type                                 | Description  | Use case examples  |
|---|--|--|
| External user <sup>*</sup><br>(authenticated) | Obtains secure access to personalized data by utilizing authentication mechanisms such as Azure AD B2C, LinkedIn, Okta, etc. | <ul style="list-style-type: none"> <li>B2B - Partner management (Dealer, Supplier, Franchise etc.)</li> <li>B2C - Account management etc.</li> </ul> |
| Anonymous user                                | Access publicly viewable web pages powered by the portal   | Knowledge management sites   |
| Internal user                                 | A user licensed with Power Apps or Dynamics 365  | Employee self-serve  |

<sup>\*</sup>External User\* means users that are not employees, onsite contractors or onsite agents of Customer or its Affiliates.

The licensing scheme varies based on end user type.

| End user type                    | Licensing model   | Description   |
|----------------------------------|---|---|
| External user<br>(authenticated) | Per login   | A login provides the authenticated user with access to a single portal for up to <b>24 hours</b> . Multiple logins during the 24-hour period count as <b>1 billable login</b> . Think of a login as a day pass to a single Power Apps Portal. |
| Anonymous user                   | Per page view   |   |
| Internal user                    | <ul style="list-style-type: none"> <li>License fee pays for access to Power Apps Portals</li> <li>No additional monetization</li> </ul> | Custom portal use rights are aligned with custom app use rights   |

### 3. Business case intro

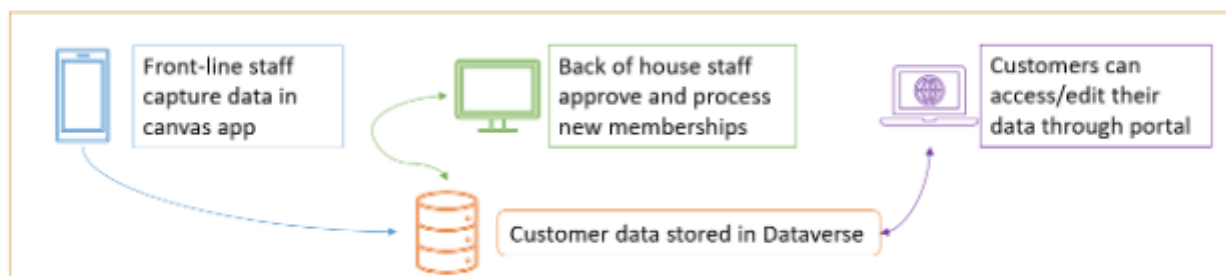
#### Nexacu Gym

- Front of house (Gym staff) use a canvas app to capture information of new gym members (Name, email, DOB, etc) and what gym membership they want to sign up for.
- Back of house (Admin staff) use a model-driven app to progress a new members application through a number of steps (payment, address confirmation, sending member pack)
- Customers can access and update their information (Address, phone) through a publicly available website
- All data is stored in the Dataverse.

#### Levels of membership:

| Member Level | Member pack | Free Training Plan | VIP pack (towel, water bottle, etc) |
|--------------|-------------|--------------------|-------------------------------------|
| Silver       | ✓           |                    |                                     |
| Gold         | ✓           | ✓                  |                                     |
| Platinum     | ✓           | ✓                  | ✓                                   |

#### Process of creating solution





## 4. Dataverse

The Dataverse is a *Database as a service*. **A built-in managed data platform.**

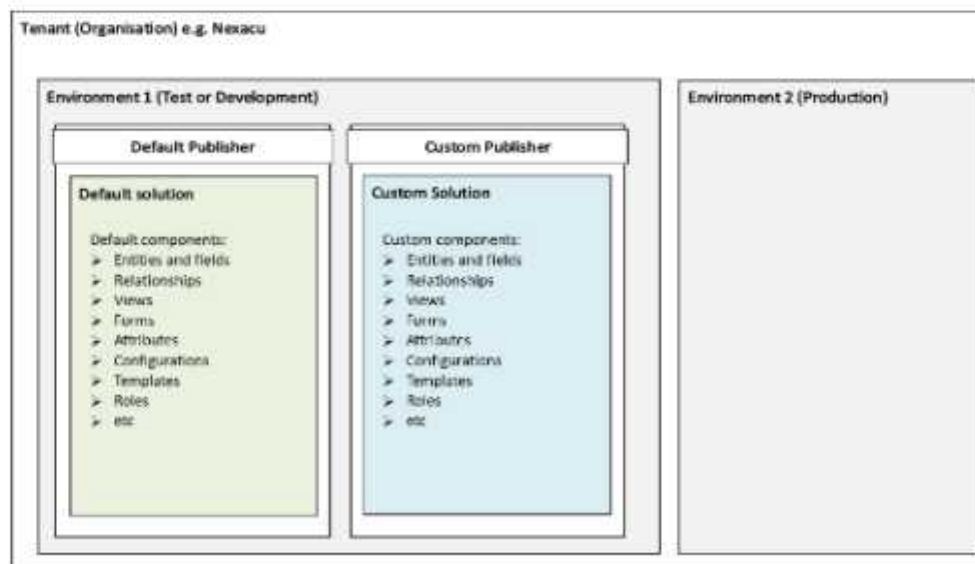
It includes everything you need to securely store, manage and utilise your data.

Under the hood, Dataverse includes Azure SQL databases, Azure blob storage, secure authentication, authorization and auditing, data validation, logic rules, and native integration across the Power Platform.



### Tenants, Environments, Publishers, Solutions, and Customisations

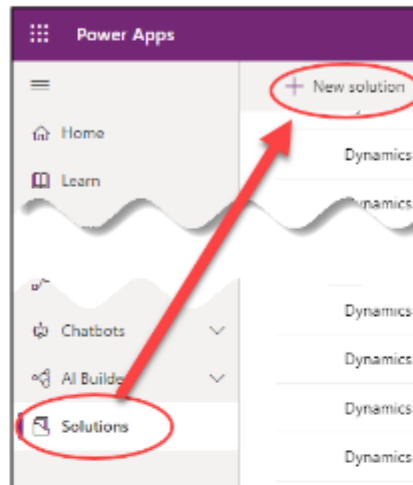
Each instance of a Dataverse is stored within an *Environment*, we can have one or many environments within our Microsoft tenant. Within an Environment we store our *components* (tables, columns, apps, flows, etc) within *Solutions*. Solutions are owned by *Publishers*, every tenant has a default publisher that creates all the standard components (user tables, roles, etc).



## Setting up Dataverse

### Ex 4.1 - Create Solution and publisher

1. Go to **make.powerapps.com**
2. Select your Power Platform **environment** (your instructor will provide this)
3. Select **Solutions** from the bottom of the left menu
4. Select **+New solution**



5. Fill in **Display name** ("MySolution")
6. Create a new **+ publisher**

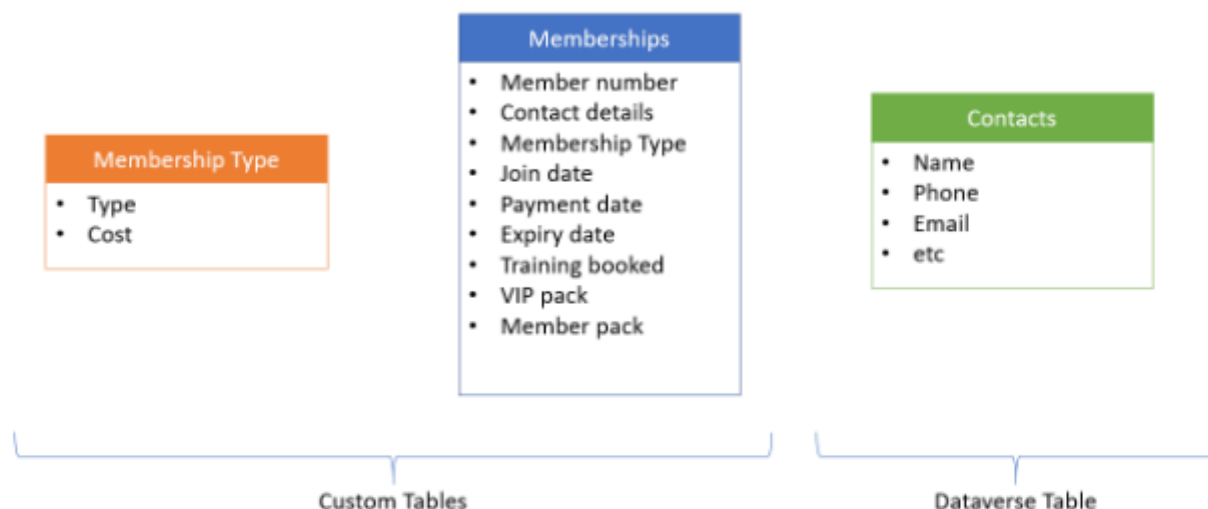
A publisher should be created for the organisation. Don't use the default.

7. Fill in **Display name** and **Prefix**. Click **Save**
8. Select your new **publisher**
9. Click **Create**

10. Open the created Solution. We'll create all customisations within this solution.

The main part of any database are the tables that it consists of, within our tables we store our records as rows. Dataverse has a number of default tables but we can also easily create new ones to suit our purpose.

We'll be using these tables in our business case:

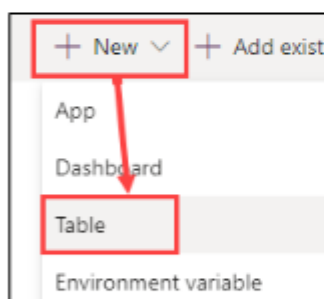


## Tables

### Ex 4.2 - How to create a new table

#### Continue from previous exercise

1. From within your solution, select **+ New**
2. Select **Table**
3. Create a new table called **Memberships**
4. Table display name: Memberships  
Primary Name Column: Member Number
5. Tick **Enable attachments**
6. Select **More Settings**
7. Expand **Collaboration**
8. Tick **Enable for activities**
9. Click **Create** (click **OK** to warnings)



A screenshot of the 'New table' form. The following fields and options are visible:

- Display name \***: Memberships
- Plural display name \***: Memberships
- Name \***: mypub\_ Memberships
- Primary Name Column**: Member Number
- Enable attachments** (including notes and files): ☒
- Enable for activities**: ☒

A screenshot of the 'Collaboration' section. The following options are visible:

- Collaboration**: Enable features to help users to more easily work together on this entity.
- Allow feedback**: ☐
- Enable for activities**: ☒

10. Review fields that are created automatically – these are mostly meta data tags  
(Created On, Modified By, etc)
11. Change the Primary Name Column to Auto number, Optional, Prefix

**Member Number** ✕

Display name \*  
Member Number

Name \* ⓘ  
mypub\_ Memberships

Data type \* ⓘ  
Autonumber

Required \* ⓘ  
Optional

☒ Searchable ⓘ

Autonumber type \* ⓘ  
String prefixed number

Prefix  
GYM

Minimum number of digits \* ⓘ  
4

Seed value \* ⓘ  
1,000

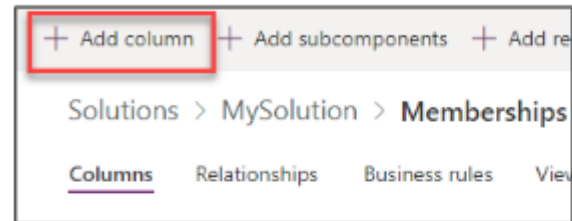
Preview ⓘ  
GYM-1000  
GYM-1001  
GYM-1002

## Columns

### Ex 4.3 - Creating columns

**Continue from previous exercise, with the Memberships table selected**

1. Select **+ Add column**



2. Enter the following:

Display Name: Join date

Data type: Date only

Required: Required

Advanced Options > Behaviour:  
change to date only

3. Click **Done**



#### Date Behaviours

**User Local:** Stores the date and time in UTC, converts the UTC to the user's time zone

**Date Only:** Stores the time value as 12am (0:00:00). No conversion. Good for birthdays and anniversaries where time isn't required.

**Time Zone Independent:** Time is stored but not converted. Good for check-in/-out times for a hotel.

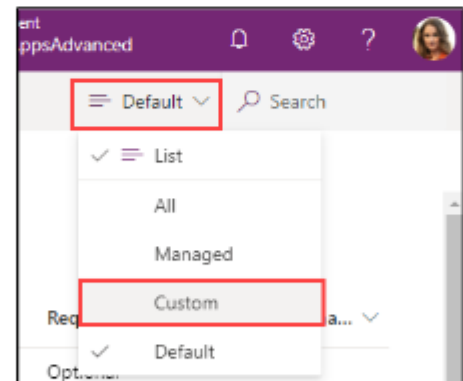
4. Create these new columns:

a. Display Name: **Expiry date**

Data type: Date only



- b. Display Name: **Payment received on**  
Data type: Date Only
  - c. Display Name: **Member pack sent**  
Data type: Yes/No
5. Create another new column  
Display Name: **Training plan booked**
6. For data type select **Choice**
7. In the Choice dropdown list, select **+ New Choice**
8. Display Name: **Yes No NA** (Remove the default name "Training plan booked")
9. Create a New option called **Yes**
10. Select **Add new item**
11. Create another New option called **No**
12. Repeat for a 3<sup>rd</sup> option **NA**
13. Click **Save**
14. In the new column pane, the new Choice option "Yes No NA" should be selected
15. Select the default value of **No**
16. Create another column  
Display Name: **VIP pack sent**  
Data type: Choice > "Yes No NA"  
Default value: No
17. Click **Done**
18. Click **Save Table!**
19. Change the column view from Default to **Custom**
20. Review the columns you have created.



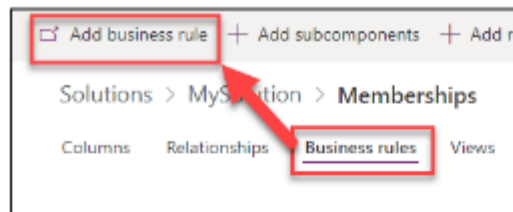
## Business Rules

Business rules and recommendations allow us to create and apply logic and validations without writing code. We can set column values, show or hide a column, as well as show error messages or recommendations.

## Ex 4.4 - Setting Business Rules

**Continue from previous exercise, with the Memberships table selected**

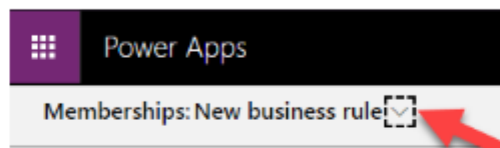
1. From the top menu select **Business rules**



2. Select **+ Add business rule**

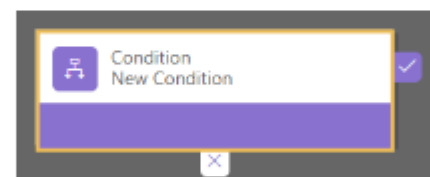
This will open a new window. If the window doesn't load close it and try again.

3. Next to Memberships: New business rule, Click the **arrow** to expand



4. Set Business rule name and description

5. Click arrow to collapse
6. Select the **Condition** box in the designer window
7. On the right-hand side select the **Properties** tab
8. Change the properties to:



Display name: **Check payment received date**

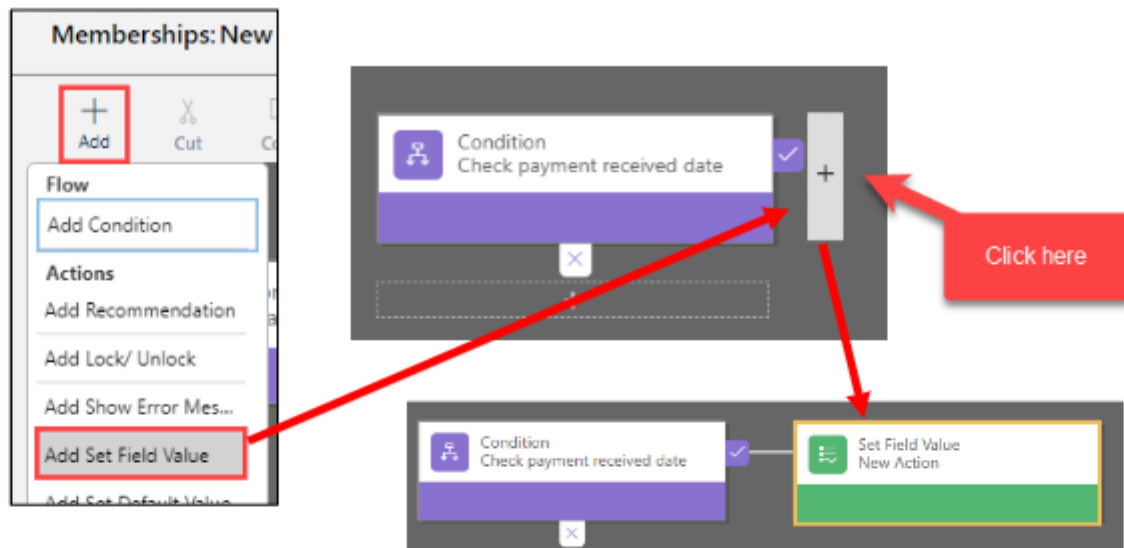
Entity (i.e. the Table): **Memberships**

Source: **Entity**

Field: **Payment received on**

Operator: **Contains data**

9. Click **Apply** (The name in the condition will update, but no other obvious signs happen)
10. In the top menu select **+Add**
11. Select **Add Set Field Value**
12. Click the area near the  
True (tick icon) of the condition.



13. On the right-hand side select the **Properties** tab

14. Change the properties to:

Display name: **Set expiry date**

Entity (i.e. the Table): **Memberships**

Field: **Expiry date**

Type: **Formula**

Field: **Join date**

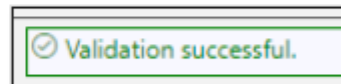
Operator: **+**

Type: **Value**

Days: **365**

15. Click **Apply**

16. Click **Validate**



17. Click **Save**

18. Click **Activate**

Processes only run when activated. To edit them you need to de-activate first

19. Close the tab and refresh the entity details

| Components      | Properties  |
|-----------------|-------------|
| Set Field Value |             |
| Display Name    |             |
| Set expiry date |             |
| Entity          |             |
| Memberships     |             |
| Field Value     |             |
| Field           | Expiry date |
| Type            | Formula     |
| Field           | Join date   |
| Operator        | +           |
| Type            | Value       |
| Days            | 365         |

## More tables

Next we'll create a new table to store details about our membership types including the name of the membership, pricing, etc.

## Ex 4.5 - Create Membership type table

### Continue from previous exercise

1. Select your **Solution**

2. Click **+ New**

3. Select **Table**

Display Name: **Membership type**

Primary name column Display name: **MemberType**

4. Click **Create**

5. Add a **column**

Display name: **Cost**

Data type: **Currency**

6. Click **Done**

7. Add a **column**

Display name: **Ex-GST**

Data type: **Currency**

8. Next to *Calculated or Rollup*, click **+Add**

9. Select **Calculation**

10. Select **Save**

A new window will try to open. Your browser might try to block it. If so, turn your pop-up blocker off then open the column again and select **Open calculation**

11. Under action, select **+Add action**

12. Find your Cost column (i.e. mypub\_cost)

13. Divided by 1.1 (type in "/1.1")

14. Click the **Tick** icon to add the action

15. Click **Save and close**

16. Refresh your table by clicking **Done**

17. Lastly, add one more column

Display name: **Icon**

Data type: **Image**

18. Click **Done**
19. Click **Save Table!**

## Adding data using Excel

The Dataverse has strong integration with Office 365 including Excel. Through Power Apps we can create a connection between the data stored in tables and Excel – allowing us to add, edit and delete records using a spreadsheet.

For this exercise you will need to be logged into the Microsoft PowerApps Office Add-in within Excel as the same user that you have created the Dataverse tables with.

### Ex 4.6 - Using Excel to add data

#### Continue from previous exercise

1. Ensure the **Membership Type** table is selected
2. From the table's menu select **Data**



3. From the top menu select Data and then select **Edit data in Excel**

This will download an Excel file with the required connections to the data already created.

4. Open the downloaded file and sign into the **Microsoft PowerApps Office Add-in** window using the same account that you have created your Dataverse with.

5. Within the Microsoft PowerApps Office Add-in select **New**  
A new row is added to the table.

6. Add the following details:

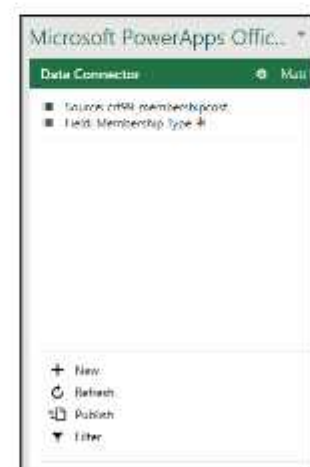
MemberType: **Silver**

Cost: **500**

7. Within the Microsoft PowerApps Office Add-in select **Publish**

You will see new items appear in the table for the Silver record, such as Status, Ex-GST, Created on, etc.

8. **Close** the Excel file





No need to save the file as the data is already published.

---

## Interact with data in the browser

Adding data using Excel is an easy way to interact with the database. But we can also view and interact with data using the browser. Our default views of the data are limiting though, so we'll have to customise them.

---

### Ex 4.7 - Creating views and forms to see and add data

#### Continue from previous exercise

1. Return to the browser and the Data view of your table
2. From the top menu select **Refresh data**

The Membership type Silver should appear but we can't see all the important columns of information.

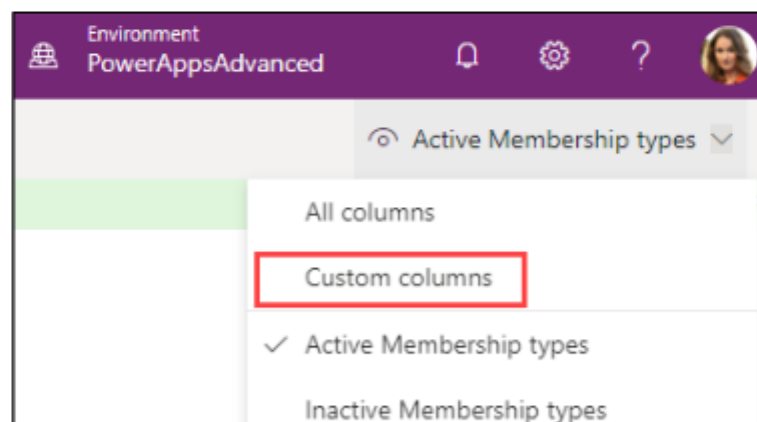
What we are looking at is called a **view**. A view defines the columns we can see and also what filters are applied to the records.

3. From the top right menu, click on **Active Membership types**

This is the current view and we can select other views from here.

4. Select **Custom columns**

This displays the data with all the columns that we have created. However, this doesn't change the columns that are shown within the *Active Membership types* view.



5. From the table's menu select **Views**
6. Select the **Active Membership types view**

This opens the view in the view designer window.

7. From the left-hand menu click and drag the **Cost** column in between the Headers for MemberType and Created On

8. From the top right hand menu, select **Publish** (this also saves the view)
9. Once the Publish button stops spinning, click the **Back** arrow in the top left-hand
10. Return to the **Data** tab to see the new updated view with the cost column
11. From the top menu, select **Add record**

This will open a new browser tab and will display a **form** with two fields (MemberType and Owner). Whereas a view shows us a list of many records, a form shows us the details of a single record. Like the default view, the default form doesn't show us a great deal of information – so we'll customise it.


12. **Close** the browser tab and return to the **Data** tab
13. You might have a notification about *Adding and editing records*, click **Okay**
14. From the table's menu select **Forms**
15. Select the **Information** form with type **Main**

Tables > Membership type

Columns Relationships Business rules Views Forms Dashboards Charts Keys Data

| Name ↑ ▾    |     | Form type ▾   | Type ▾ |
|-------------|-----|---------------|--------|
| Information |     | Card          | Custom |
| Information | ... | Main          | Custom |
| Information | ... | QuickViewForm | Custom |

This will open the form designer

16. In the top left-hand corner click the Hamburger icon 
17. Select **Table columns** from the left-hand menu (It's the icon with *Abc* written in a box)
18. Click and drag the **Cost** column onto the form, just under MemberType
19. Repeat for the **Icon** column
20. On the form, click the **Owner** field
21. In the properties pane on the right, tick **Hide**
22. Click **Publish**
23. After publishing, click the **Back** arrow

If your Back arrow disappears return to [make.powerapps.com](https://make.powerapps.com)

24. Return to the **Data** tab of the Membership Type table
25. Mouse over the **Silver** record and click anywhere to select it

26. From the top menu select **Edit record**
27. In the new browser tab, next to Icon select **Choose File**
28. From the Power Apps Exercises > Icons, select **Silver icon.png**
29. From the top menu, select **Save & Close**
30. In the Data tab, select **Add record**
31. Create a **Gold** membership type, cost of \$1000
32. Click **Save**
33. Upload the **Gold icon.png** image file
34. Click **Save & Close**
35. Repeat steps 30-34 for Platinum
  - MemberType: **Platinum**
  - Cost: **1500**
  - Icon: **Platinum icon.png**
36. From the top menu, select **Refresh data**

|  |               |                |       |       |            |        |      |
|--|---------------|----------------|-------|-------|------------|--------|------|
| Solutions > MySolution > Membership type |               |                |       |       |            |        |      |
| Columns                                  | Relationships | Business rules | Views | Forms | Dashboards | Charts | Keys |
| <b>Data</b>                              |               |                |       |       |            |        |      |
| MemberType                               |               | Cost           |       |       |            |        |      |
| Gold                                     |               | \$1,000.00     |       |       |            |        |      |
| Platinum                                 |               | \$1,500.00     |       |       |            |        |      |
| Silver                                   |               | \$500.00       |       |       |            |        |      |

---

## Relationships

Like any relational database, relationships between tables are a essential part to creating an efficient data schema. We are going to create relationships that will allow us to assign a contact (i.e. a person) and a membership type to a membership record.

## Ex 4.8 - Creating Relationships

### Continue from previous exercise

1. Browse to your Solution (use the breadcrumb navigation at the top of the page)
2. Open the Memberships table
3. Select the **Relationships** tab
4. Review the existing relationships that are created by default between table

For instance, there are a number of relationships between the Memberships and User tables to assist in identifying who created, modified or owns a record.

5. From the top menu select **Add relationship**
6. Select **Many-to-one**
7. Under the Related (One) Table select **Membership Type**

Each membership record can only be one membership type. While each membership type is related to many memberships, therefore many-to-one relationship.

8. Rename the **Lookup column display name** to **MembershipTypeLU**

The LU stands for Look Up and will help us find it later on.

9. Click **Done**
10. Create another relationship between Memberships and the Contact.

Relationship type: **Many-to-one**

Lookup column display name: **ContactLU**

11. Click **Done**
12. Click **Save Table**

**Many-to-one**

Choose the **Related entity** to which to create your relationship lookup. [Learn more](#)

| Current (Many)              | Related (One)   |
|-----------------------------|-----------------|
| Entity *                    | Entity *        |
| Memberships                 | Membership Type |
| Lookup field display name * |                 |
| MembershipTypeLU            |                 |
| Lookup field name *         |                 |
| mattb_MembershipTypeLU      |                 |

## More Business Rules

Now that we have our tables and relationships set up, we can now create our last business rule.

The purpose of second rule is to set what membership perks someone is entitled to based on their membership type. If a member isn't entitled to a perk we will set it to N/A and hide the field.

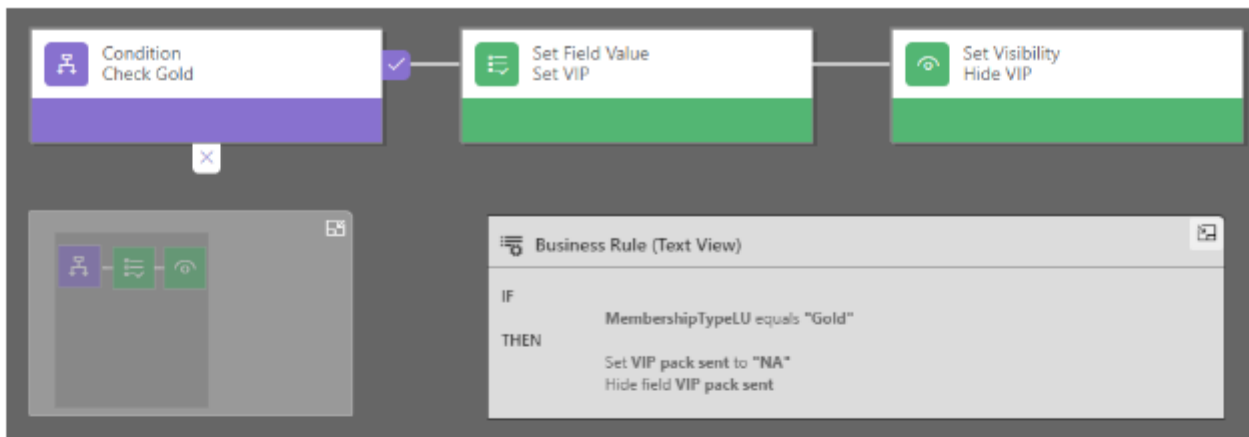
| Member Level | Member pack | Free Training Plan | VIP pack (towel, water bottle, etc) |
|--------------|-------------|--------------------|-------------------------------------|
| Silver       | ✓           | ✗                  | ✗                                   |
| Gold         | ✓           | ✓                  | ✗                                   |
| Platinum     | ✓           | ✓                  | ✓                                   |

---

**Ex 4.9 - Create 2nd business rule****Continue from previous exercise**

1. Within the Memberships table, select the **Business rules** tab
2. Select **Add business rule**
3. Change the business rule name to **Set membership perks**
4. Select the first **Condition**
5. Change properties to:  
Display name: **Check Gold**  
Rule 1 Field: **MembershipTypeLU**  
Rule 1 Value: Click the **magnifying glass** icon, select **Gold**, select **Add**
6. Click **Apply**
7. **Add a Add set field value** action to the true (tick) outcome of the Check Gold condition
8. Change properties to:  
Display Name: **Set VIP**  
Field: **VIP pack sent**  
Value: **NA**
9. Click **Apply**
10. **Add a Add set visibility** action to the right of the Set VIP action
11. Change properties to:  
Display Name: **Hide VIP**  
Field: **VIP pack sent**  
Visible: **No**
12. Click **Apply**





13. **Add a Add condition** to the false (cross) outcome of the Check Gold condition

14. Change properties to:

Display name: **Check Silver**

Rule 1 Field: MembershipTypeLU

Rule 1 Value: Click the **magnifying glass** icon, select **Silver**, select **Add**

15. Click **Apply**

16. Select the **Set VIP** action

17. Click **Copy** from the top menu

18. Click **Paste** from the top menu

19. Select the Plus area to the right of the *Check Silver* condition

20. Select the **Hide VIP** action

21. Click **Copy** from the top menu

22. Click **Paste** from the top menu

23. Select the Plus area to the right of the *Set VIP* action

24. **Add a Add set field value** action to the right of the new *Hide VIP* action

25. Change properties to:

Display Name: **Set training plan**

Field: **Training plan booked**

Value: **NA**

26. Click **Apply**

27. **Add a Add set visibility** action to the right of the Set VIP action

28. Change properties to:

Display Name: **Hide training plan**

Field: **Training plan booked**

Visible: **No**

29. Click **Apply**

30. From the top right menu click **Validate**

If the validation isn't successful check your steps and make sure you applied the changes to the properties of each action and condition.

31. Click **Save**

32. Click **Activate** and confirm the activation



s Rule (text view)

MembershipTypeLU equals "Gold"

Set VIP pack sent to "NA"  
Hide field VIP pack sent

MembershipTypeLU equals "Silver"

Set VIP pack sent to "NA"  
Hide field VIP pack sent  
Set Training plan booked to "NA"  
Hide field Training plan booked

2021

[Back to top](#)

Power Apps canvas

## 5. Power Apps canvas



So far in our solution we have created a number of custom tables with custom

## 5. Power Apps canvas



So far in our solution we have created a number of custom tables with custom columns. A solution can contain many types of artifacts including flows, canvas apps, and model-driven apps.

In the next exercise we are going to import a pre-built canvas app into our solution. Currently we can't import it straight into the solution so we need to first import it into the default solution and then add it into ours.

We'll then customise the app including:

- displaying the current version number of the app
- connecting to our Dataverse tables
- creating input forms for our new Gym members
- taking a look at the barcode reader control

### Importing an app

---

#### Ex 5.1 - Importing an App

##### Continue from previous exercise

1. Within make.powerapps.com select **Apps** from the left-hand menu
  2. From the top menu select **Import canvas app**
  3. Click **Upload** and select the file **GymMembershipCanvasApp.zip**
  4. Check that the Import Setup is **Create as new**  
If it says Update, click the word Update and change to Create as new
  5. Click **Import**
  6. From the left menu select **Solution**, then select your Solution (e.g. MySolution)
  7. From the top menu select **Add existing > App > Canvas app**
  8. Select **Outside solutions**
  9. Select the **Gym membership** app
  10. Click **Add**
-

## Displaying the app version within an app

This is a simple and effective way to display the version of the app in the app itself.

### Ex 5.2 - Display App Version

#### Continue from previous exercise

1. Click on the **3 dots** (the ellipsis) of the canvas app, Gym Membership
2. Select **Details**
3. Highlight and **copy the App ID** GUID  
A GUID is a Globally Unique Identifier, this is a 128-bit integer number used to identify resources.
4. From the top menu select **Edit** to open the app in create.powerapps.com
5. Click **Allow** to enable the Power Apps for Makers and Office 365 Users connectors
6. Make sure the **HomeScreen** is selected
7. In the properties drop down select **OnVisible**
8. Expand the formula bar
9. In the formula replace the text **!!!Insert App URL Here!!** with the copied App ID GUID  
The GUID should be enclosed by double apostrophes
10. Select a different screen and then navigate back to the HomeScreen.

This will trigger the OnVisible event and update the collections

#### OPTIONAL

11. From the top menu select **View > Collections**
12. Open the **MyAppVersion1** collection  
This is the initial collection where we store all of the apps details through the PowerAppsforMakers.GetAppVersions(AppID) function. You'll notice that it is just one record.
13. Click the record icon to view the items within the record
14. From the collections menu select **UngroupedVersions**  
This is the collection where we have performed a Ungroup of the first collection, exposing the next level of the record.
15. Click on the record icon in the properties column for the bottom most row  
This will show the properties for this row including the lifeCycleId – which is where we look for "Published" before returning the corresponding version and date.



16. Click the **back arrow** to return to the Power Apps designer

---

## Connect to the Dataverse

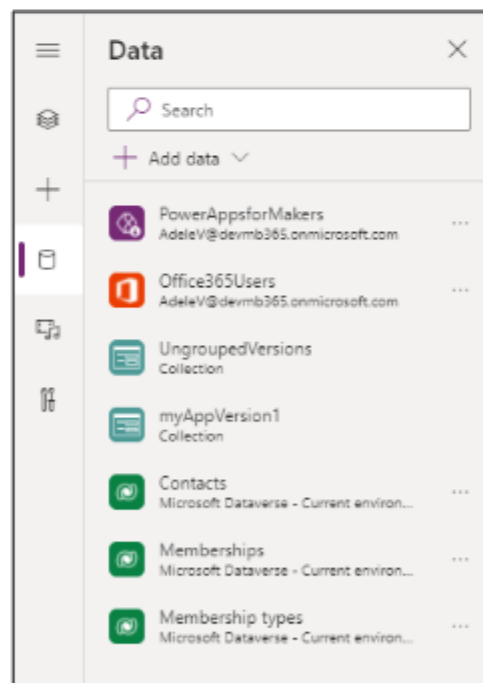
Since the Dataverse is already apart of the Power Platform connecting to our tables is very easy.

---

### Ex 5.3 - Connect canvas app to Dataverse tables

#### Continue from previous exercise

1. From the left-hand menu select the **Data** icon (the small cylinder)
2. Click **Add data**
3. Select **Contacts**
4. Click **Add data** again
5. In the search bar type **member**
6. Select **Memberships**
7. Repeat steps 4-5 and select **Membership Type**



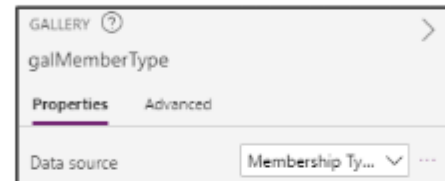
## Connect a gallery to the Dataverse

### Ex 5.4 - Connect a gallery to the Dataverse

#### Continue from previous exercise

1. Select the **AddScreen**
2. Select the gallery **galMemberType**
3. From the Select a data source pane, select **Membership Types**
4. Alternatively, in the gallery's properties select Membership Types in the Data source property drop down

The Membership types details will populate the gallery as the controls are all pre-configured.



**Note:** If the images disappear go to the left-menu, select Data, then click the 3-dots next to Membership types and select Refresh



## Connect forms to the Dataverse

### Ex 5.5 - Use forms to add data to the Dataverse

#### Continue from previous exercise

1. Insert 2 edit forms on the AddScreen (**Insert > Forms > Edit**)
2. With the two new forms, rename and connect them to Dataverse tables using the following names and data sources:

| Default name | New Name       | Data source |
|--------------|----------------|-------------|
| Form1        | formAddContact | Contacts    |
| Form2        | formAddMember  | Memberships |

- Place formAddContact on the top half of the right-hand white square and the second form on the bottom half.

With **formAddMember**:

- Select formAddMember
- In the properties pane click **Edit fields**
- Select the following fields:
  - ContactLU
  - Join date
  - MembershipTypeLU
- Remove any pre-populated fields

8. Select the **ContactLU** card
9. In the properties pane, select **Advanced**
10. Click **Unlock to change properties**
11. Select **Default** property
12. Change formula to: **formAddContact.LastSubmit**
13. Unlock the **Join date** card
14. Change Default property formula to: **Today()**
15. Unlock the **MembershipTypeLU** card
16. Change Default property formula to: **galMemberType.Selected**

With **formAddContact**

17. Add fields (First name, Last name, Email, Mobile Phone, Birthday)  
Remove other pre-populated fields
18. Set Hint Text for name fields (Select the label component to set this property)
19. Set form to 1 column, horizontal layout

## Submitting the data

Instead of writing `Patch()` functions to add new entries to our tables we are going to use the forms. Thinking about the relationships between our Dataverse tables, we first need to create the Contact record before creating an associated Memberships record. Therefore we'll submit the Contact form first and then use the `LastSubmit` action to associate the new Contact record with the new Membership record.

The steps will be:

1. When we click the tick icon, we will submit the **formAddContact**. This will create a record in the contact entity.
2. After the first form is submitted the `ContactLU` field is populated with the newly created contact record, then the `formAddMember` is submitted.
3. After that is submitted successfully, we are taken back to the Home Screen.

---

### Ex 5.6 - Submit the data

#### Continue from previous exercise

1. Set `formAddContact OnSuccess: SubmitForm(formAddMember)`
2. Set `formAddMember OnSuccess: Navigate(HomeScreen, ScreenTransition.None)`
3. Set `formAddMember visibility` to false
4. Preview the app by pressing **F5**

You might need to navigate to the home screen and click **Add new member**

5. Submit two new members
6. Review the data in the tables to see if it all worked

| Tables > Memberships           |   |
|--------------------------------|---|
| Columns                        | Relationships Business rules Views Forms Dashboards Charts Keys <u>Data</u> |
| Member Number                  | Created On  |
| GYM-1000                       | 6/22/2021 4:51 AM   |
| <input type="radio"/> GYM-1001 | 6/22/2021 4:51 AM   |

| Tables > Contact |   |
|------------------|---|
| Columns          | Relationships Business rules Views Forms Dashboards Charts Keys <u>Data</u> |
| Full Name        | Email   |
| Jerry Mouse      | Jerry@catsdrool.com   |
| Tom Cat          | Tom@catsrule.com  |

## Using data tables

The following two exercises are optional exercise that your trainer may run through, otherwise you can complete after the class.

### Ex 5.7 - Inserting a data table

#### Continue from previous exercise

1. Go to the **FindScreen**
2. Insert a **Data table**
3. Connect the Data table to the **Memberships** table
4. Add Fields:
  - Member number
  - Contact (change text property of column to 'Full Name')
  - Join Date
  - Membership Type (Change text property of column to MemberType)
5. To update headers change FieldDisplayName property:
  - ContactLU to Full Name
  - MembershipTypeLU to Member Level
6. Set the combo box
  - a. Data source: Contacts
  - b. Fields: Full name, Mobile Number
  - c. Search field: mobile number

| Member Number | First Name | Membership Level | Join date  |
|---------------|------------|------------------|------------|
| GYM-1000      | Tom        | Silver           | 22/05/2021 |
| GYM-1001      | Jerry      | Gold             | 22/05/2021 |



## Filtering data tables

---

### Ex 5.8 - Filter data table

#### Continue from previous exercise

1. First we'll filter based on the selected items in the combo box.

On the Data table, change the **Items** property to:

```
Filter(  
    Memberships,  
    ContactLU.'Full Name' = ComboBox1.Selected.'Full Name'  
)
```

2. Preview the app and search for a mobile number. Remember that we selected to show the first name but search on the mobile number.

When you select the fields of the combo box you can also select the layout, choosing a double layout will show two fields. But you can still only search on one.

3. Next we'll alter the filter based on if we have clicked on the combobox or the barcode reader.

This uses the OnSelect property of those components and a variable – varSearch.

The new parts of the formula are highlighted.

Change the Data Table Items to;

```
If(  
    varSearch,  
    Filter(  
        Memberships,  
        ContactLU.'Full Name' = ComboBox1.Selected.'Full Name'  
    ),  
    Filter(  
        Memberships,  
        'Member Number' = varBarcode  
    )  
)
```

What if the combobox is empty?

4. Add the If IsEmpty functions shown below.

If the combobox is empty the gallery will display all records.

**Note:** Don't forget the extra brackets

```
If(
    varSearch,
    If(
        IsEmpty(ComboBox1.SelectedItems),
        Memberships,
        Filter(
            Memberships,
            ContactLU.'Full Name' = ComboBox1.Selected.'Full Name'
        )
    ),
    Filter(
        Memberships,
        'Member Number' = varBarcode
    )
)
```

You can test out the Barcode scanner using the Power Apps mobile app.

---

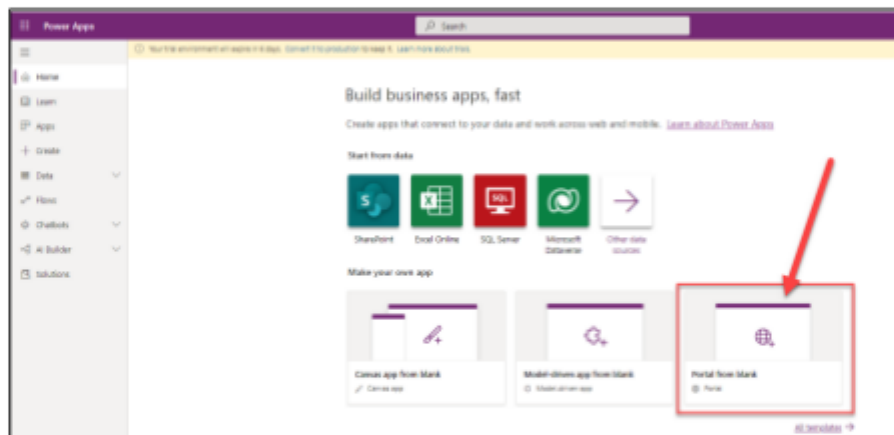
## 6. Prepare for Portal exercises

Later in the course we are going to build our customer facing website using a portal app. Portal apps can take some time to be created, so we will start the provisioning process now and then return to the app in chapter 12.

### Creating a Portal app

#### Ex 6.1 - Creating a portal app

1. Go to <https://make.powerapps.com/>
2. Select your environment
3. From the Home page click **Portal from blank**



4. Name: **Gym Portal**  
Address: **NexacuGym[First name][Last initial]**  
i.e. NexacuGymAdeleV  
Language: **English**
5. Click **Create**

It can take 10-15 minutes or longer for the portal app to be created

TIP: We'd need to install portal solution packages on this environment, and this may take a while. To try portals quickly, use an environment with the required packages already installed or create a new environment using the link [Create new environment](#).

**Name \***

**Address \*** ⓘ

☒ NexacuGymAdeleV.powerappsportals.com

**Language**

By clicking on Create, you agree to the [Terms and Conditions](#) and the [Terms of Service](#).  
[Privacy and Cookies](#)

## 7. Model-driven apps



The model-driven apps are built by composing multiple page types and components using several designers.

Additionally, there are designers for the entity and business logic. The page types come from the View Designer, Form Designer, and Dashboard Designer. Visual components include the Sitemap Designer and Business Process Flow Designer.

The App Designer then composes the app by identifying the UI elements to show. The multiple designers allow rich targeted definition of different parts of the app and its behaviour.

- App Designer specifies the sitemap, global dashboards, business processes flows, and entity forms, views, and dashboards
- Sitemap Designer provides the application navigation that is always available
- Business Process Designer provides stages and steps to guide users consistently through common business processes within a form
- Entity Designer defines the fields, relationships, and metadata for an entity
- Business Rule Designer provides no-low business logic for an entity
- View Designer specifies columns and filter conditions for a record list
- Form Designer specifies the fields and controls along with layout for a single record
- Dashboard Designer summaries one or more entities using charts, lists, etc.

The model-driven apps are fully responsive so a single definition works from web to tablet to mobile devices. This is a different with the canvas apps which the user needs to choose mobile or tablet before designing it.

### Create a model-driven app

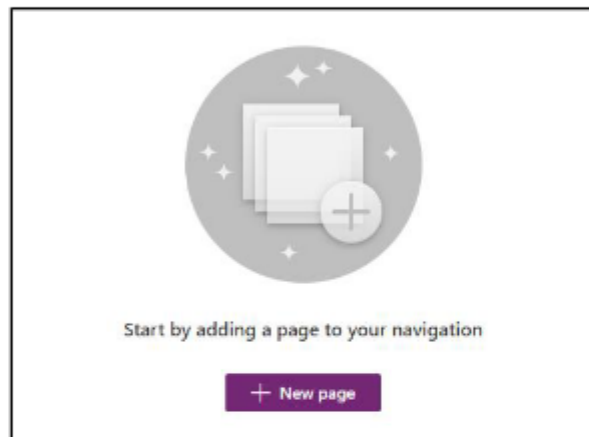
As of writing (July 2021), Microsoft have started moving to a new Power Apps designing experience. Model-driven apps will eventually be designed in the same modern experience used for canvas apps.

Over the next 12-months, Microsoft will transition to the new design experience starting with a number of preview features.

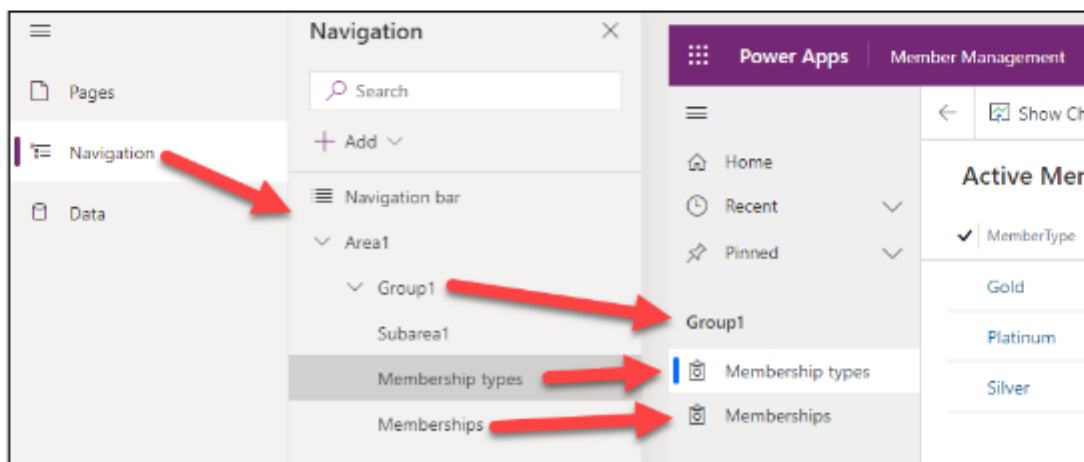
## Ex 7.1 - Creating a model-driven app

### Continue from previous exercise

1. Browse to your **Solution**
2. Select **New > App > Model-driven app**
3. Select the **Modern app designer (preview)**, click **Create**
4. Name the Model-driven app **Member Management**, click **Create**
5. On the centre of the app screen click **New Page**

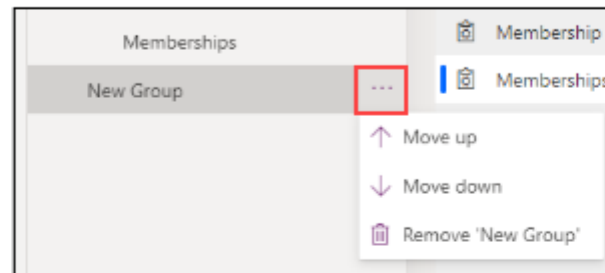


6. Select **Table based view and form**, click **Next**
7. Search for **member** and tick the box next to Membership Type and Memberships
8. Leave the box next to **Show in navigation** ticked
9. Click **Add**
10. Click **Navigation** on the left menu
11. Review the items: Area1, Group1, Subarea1

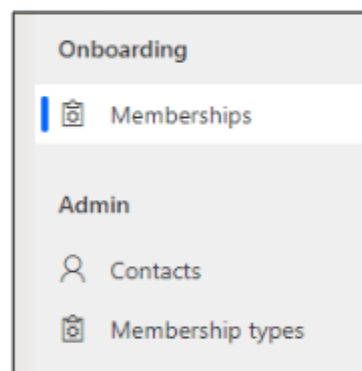


12. Select **Subarea1** from the Navigation menu
13. In the right pane, select **Table** for content type

14. Select the **Contact** table
15. Rename Subarea1 to **Contacts**
16. Select **Group1** and rename to **Admin**
17. From the navigation pane, select **Add > Group**
18. Click the 3 dots next to **New Group**, select **Move up**



19. Select **New Group** and rename to **Onboarding**
20. Move **Membership** up into the Onboarding group



21. Click **Save** in the top right to save your changes

## Create a new view

Views are used to display a list of rows, think of them like a table in Excel. We can apply filters and also customise what columns are displayed.

Earlier we modified default view to show more columns of our data. Now we want to show the administration team the membership records that haven't received a membership pack yet. These are the records they need to process so we'll create a new view for this purpose.

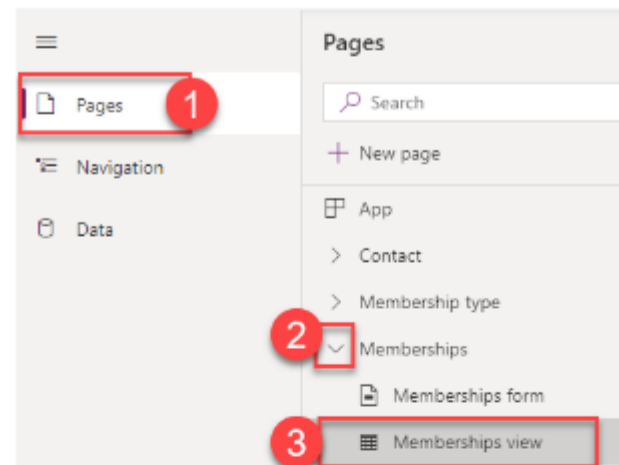
Views are created and customised for each table, so we first select the table before creating



## Ex 7.2 - Creating a new view

### Continue from previous exercise

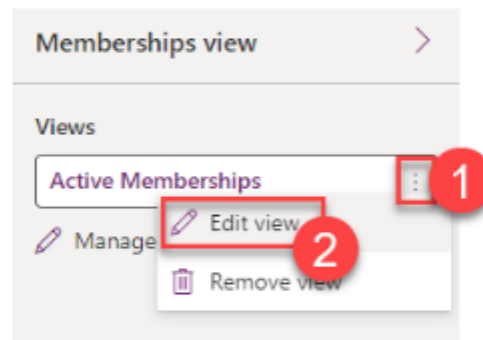
1. From the left-hand menu, select **Pages**
2. Expand the **Memberships** page
3. Select **Memberships view**



4. From the right-hand Memberships view pane select **Manage views**
5. Tick the **Active Memberships** view

As of writing there was no **New form** option in this menu. If this appears you can select this option and skip ahead.

6. Click the **3-dots** next to Active Memberships and select **Edit view**



This will open a new window with the View editor

7. In the top right, click the down arrow next to **Save**, select **Save As**
8. Name: **Membership packs not sent**
9. Click **Save**

This might reload the page.

10. From the left **Table columns** pane click and drag **Join date** on to the view. You'll need to drag it to the views header, in between *Member Number* and *Created On*.
11. On the Table Columns pane, click **Related**

This shows columns from other tables, that are related to this table.

12. Expand the **ContactLU** item
13. In the search bar type **Full Name**

14. Repeat for **Email** and **Mobile Phone**
15. In the Table Column pane expand **MembershipTypeLU**
16. Drag **MemberType** into the view
17. Right click the **Created On** column, select **Remove**

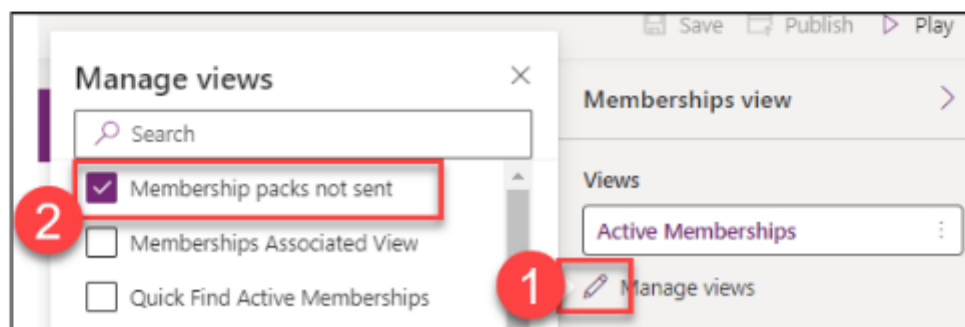
| Member Number ↑ | Join date | Full Name ... | Email (Con...      | Mobile Phone... | MemberTy... |
|-----------------|-----------|---------------|--------------------|-----------------|-------------|
| GYM-1000        | 6/22/2021 | Tom Cat       | Tom@catsrule.co... | 0412345678      | Silver      |
| GYM-1001        | 6/22/2021 | Jerry Mouse   | Jerry@catsdrool... | 0487654321      | Gold        |

18. On the right **View** pane, select **Edit Filters**
19. Click **Add**, select **Add row**
20. From the first drop down select **Member pack sent**
21. From the third drop down select **No**

This will now only display records that are active and where the Member pack sent field shows No.

22. Click **Ok**
23. In the top right corner click **Publish**

This saves and publishes the view.
24. You can **close** the view editor browser tab
25. Go back to the browser tab with your app designer
26. Click **Publish**
27. **Refresh** the browser tab
28. Navigate to the **Memberships view** again (Pages>Memberships>Memberships View)
29. Click **Manage views**
30. Select the new **Membership packs not sent** view



31. Click **Publish**

32. Click **Play** to preview the app in a new window

33. Click the name of the view **Active Memberships**

This is how you switch between views in the app.

34. Select the **Membership packs not sent** view

If it doesn't appear, refresh your browser

35. **Close** the app preview

---

## Forms

While a View shows us a list of records, a Form shows us further details of an individual record. Just like Views we can create a Form from scratch or we can modify existing forms.

In the next exercise we will modify an existing form to show the Member's contact information as well as details of their membership. This will use related information that's stored in another table.

---

### Ex 7.3 - Modify a form

#### Continue from previous exercise

1. From the Pages menu pane, select **Memberships > Memberships form**

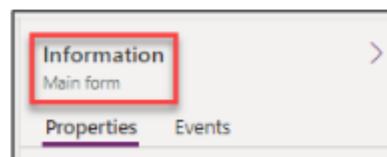
2. From the right-hand forms pane click **Manage forms**

3. Tick the last **Information** form

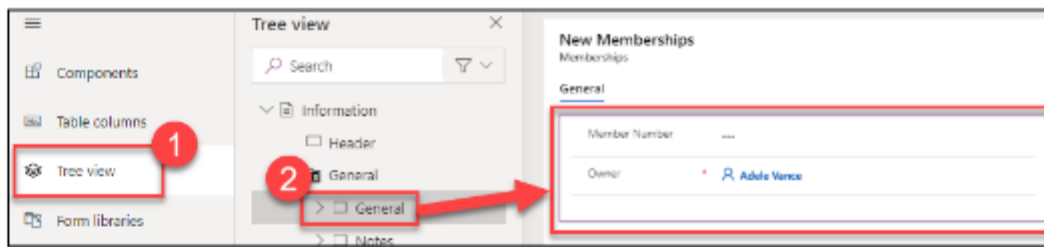
This is the tricky part as the different forms all have the same name. We want to find the *Main* form so you might need to complete these steps a couple times to get the right one.

4. Click the **3-dots** next to Information and select **Edit form**

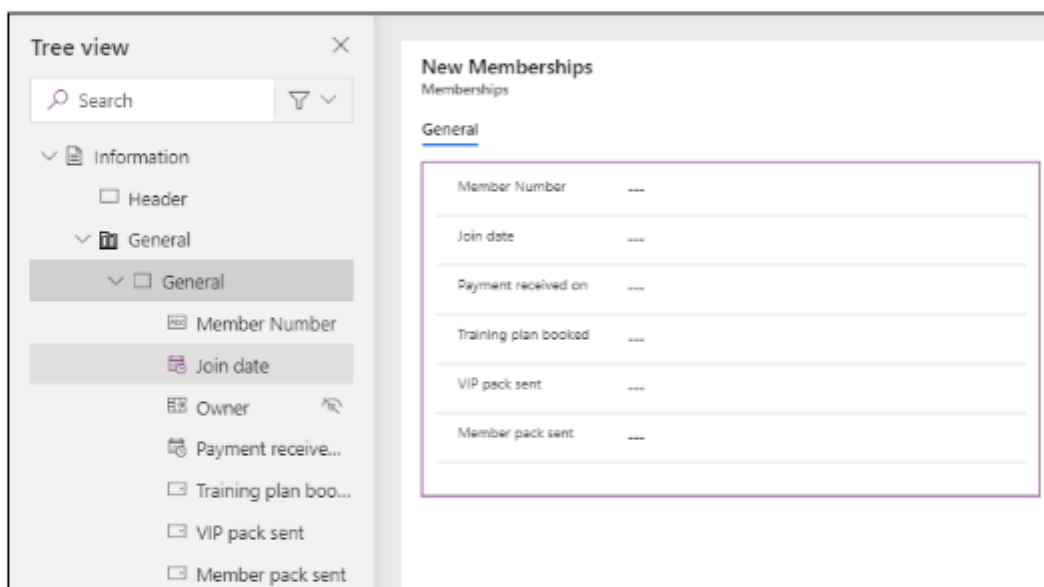
5. In the new browser tab check that you have the **Main form** open, if not return to step 3 and select a different Information form.



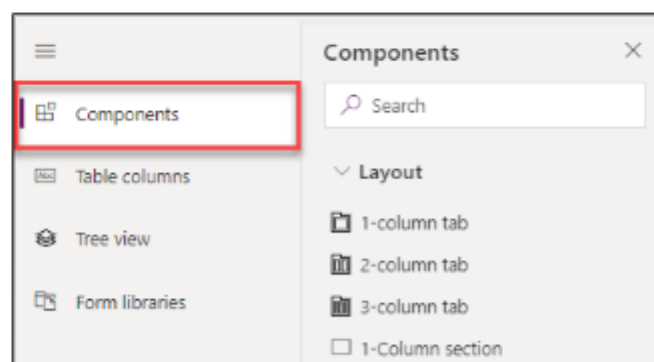
6. Select **Tree view** and expand Information > General, selecting General



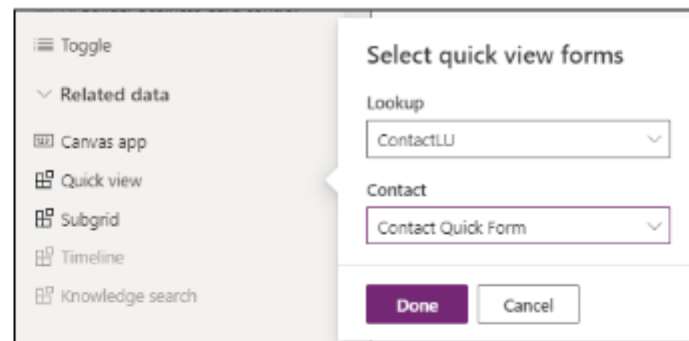
7. In the left-hand navigation pane, select **Table columns**
8. Click and drag the following columns into the form: Join date, Payment received on, Training Plan Booked, VIP pack sent, and Member pack sent
9. Hide the Owner column by first selecting it, then in the right-hand pane tick **Hide**



10. In the left-hand navigation pane, select **Components**



11. From the Components pane select **Quick view**
12. Select **ContactLU** in Lookup and **Contact Quick Form** in Contact
13. Click **Done**



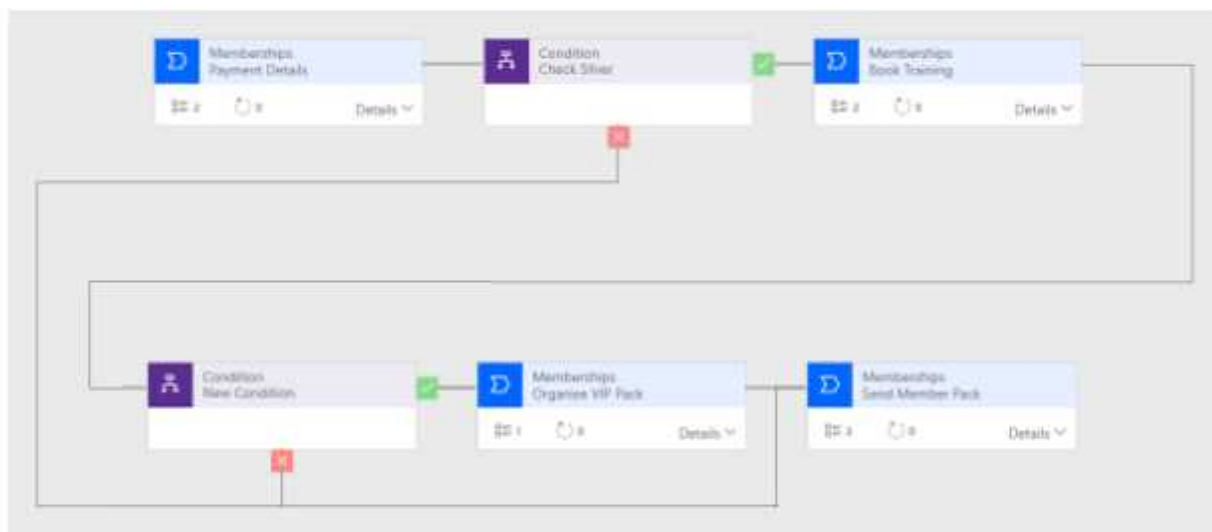
14. Repeat steps 11-13 and add in the **MembershipsTypeLU / Information** Quick form
  15. Click **Publish**, then close the window
  16. Return to your app, click **Publish**
  17. **Play** your app, select a record to see it open up in the new form.
-

## 8. Business Process Flows

With model-driven apps users can define a standard way of doing things. For instance, when a new member joins the gym, we need to check that their fees have been paid, book in their complimentary training session and then determine what will be sent in their membership pack.

These steps can be formalised using a Business Process Flow, meaning that for each new member record we need to complete the defined steps. This provides a consistent, repeatable process for all of the gym's administration staff. We'll also make the process dynamic based on the level of membership as there are more steps to complete for our VIP members.

This is what we will create in the next exercise:



### Using Power Automate

In earlier versions of the Power Platform (i.e. Dynamics CRM) Business Process Flows were created from the site map of the model-driven app or through the advanced settings (these are now known as the Classic experience). We now create the BPF using Power Automate

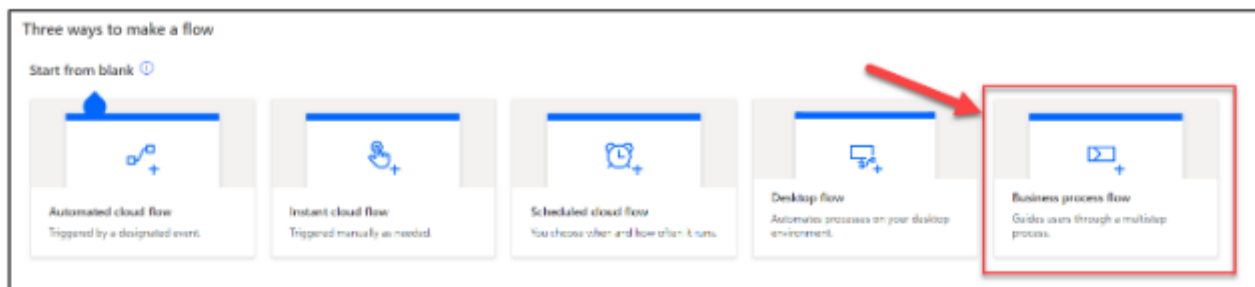
#### Ex 8.1 - Creating a Business Process Flow

##### Continue from previous exercise

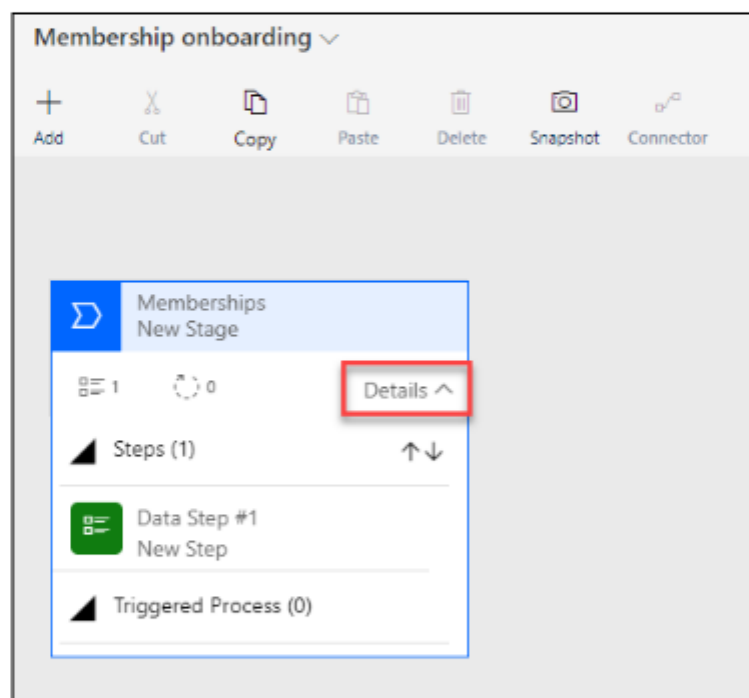
1. Open a new browser and go to [office.com](https://office.com) and open **Power Automate**  
Note: You can also go directly to [australia.flow.microsoft.com](https://australia.flow.microsoft.com)
2. In the top right corner make sure the correct Environment is selected.
3. On the left hand menu click **Create**



- From the Start from blank area, click on **Business Process Flow**

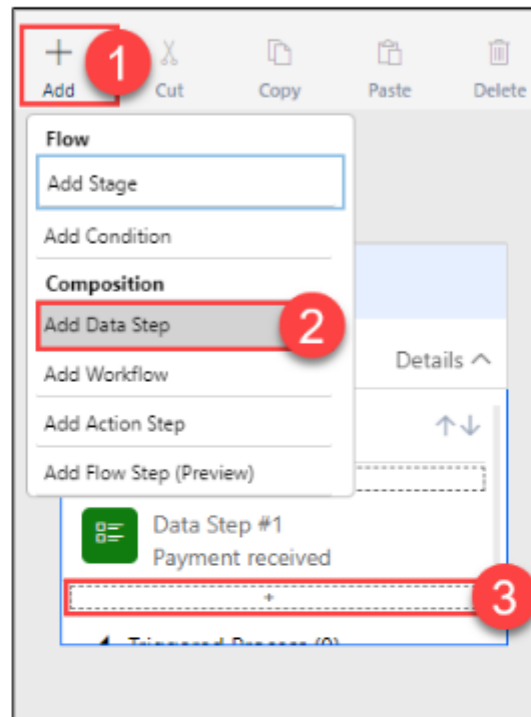


- Name the flow **Membership onboarding**  
Leave the default Name field as is
- In the Choose a table box, search for **Memberships**
- Select the **Memberships table**
- Click **Create**
- Click on the **Details** button to expand the first stage



- In the right-hand properties menu rename the stage to **Payment details**
- Click **Apply** in the bottom right corner
- Expand the **Details** section again
- Select **Data Step #1**
- In the right-hand properties menu rename the Data step to **Payment received**
- In the Data field dropdown select **Payment received on** column
- Tick the **Required** option

17. Click **Apply** in the bottom right corner
18. From the top menu click **Add**, select **Add Data Step**
19. Select the area below the data step Payment Received



20. Change the new data steps properties to:

Step name: **Confirm membership type**

Data field: **MembershipTypeLU**

21. Click **Apply**
22. On the right-hand menu select the **Components** tab

You can either use this menu or Add from the top menu bar to insert new components. They do the same thing. The only difference is that the properties pane is click-and-drag.

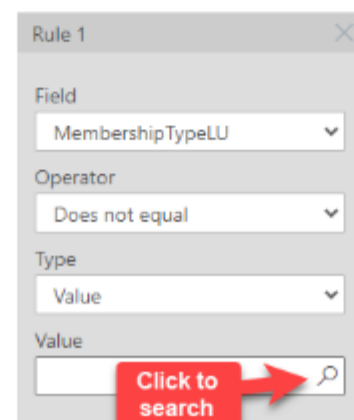
23. Click and drag the **Condition** component to the right of the first stage
24. Rename the condition to **Check Silver**
25. Under the rules make the following selections:

Field: **MembershipTypeLU**

Operator: **Does not equal**

Type: **Value**

26. In the Value field, click the magnifying glass icon
27. Select **Silver** and click **Add**



28. Click **Apply**
29. Insert a new **Stage** after the Check silver condition
30. Change Display name to **Book training**
31. Click **Apply**
32. Change the properties of the Data step to:
  - Step name: **Training plan booked**
  - Data field: **Training plan booked**
  - Tick the required option
33. Click **Apply**
34. Insert a new **Data step** and change it's properties to:
  - Step name: **Membership type**
  - Data field: **MembershipTypeLU**
  - Note: In order to use a field in a condition, it needs to be in the preceding stage.  
That's why we show it in a couple of places.
35. Insert a new **Condition**
36. Rename the condition to **Check Platinum**
37. Under the rules make the following selections:
  - Field: **MembershipTypeLU**
  - Operator: **Equals**
  - Type: **Value**
  - Value: **Platinum**
38. Click **Apply**
39. Insert a new **Stage** after the Check Platinum condition
40. Change Display name to **Organise VIP Pack**
41. Click **Apply**
42. Change the properties of the Data step to:
  - Step name: **VIP Pack Sent**
  - Data field: **VIP Pack Sent**
  - Tick the required option
43. Click **Apply**
44. Insert a new **Stage** after the Organise VIP Pack stage

45. Change Display name to **Send member pack**
46. Click **Apply**
47. Change the properties of the Data step to:
  - Step name: **Member pack sent**
  - Data field: **Member pack sent**
  - Tick the required option
48. Click **Apply**
49. Select the second condition, **Check Platinum**
50. Click **Connector** from the top menu, then select **Connect**
51. Click the last stage, **Send member pack**
52. Select the first condition, **Check silver**
53. Click **Connector** from the top menu, then select **Connect**
54. Click the last stage, **Send member pack**
55. From the top right menu, click **Validate**
  - If you get a validation error, review and rectify any items.
56. Click **Save**
57. Click **Activate**, in the pop-up window confirm the activation
  - Once activated the Business Process Flow can't be edited or deleted, you will need to first deactivate the flow.
58. Close the Business Process Flow designer tab
59. Return to the **Power Automate** tab
60. Select **My flows** from the left menu
61. Click the **Business process flows** tab
  - If you can't see your flow, check the Environment
62. Click the 3-dots on the flow, and check the Turn off option is available – this is the equivalent of the Activate button we clicked earlier.



63. Return to your **PowerApps browser** tab

Or you can go to [make.powerapps.com](https://make.powerapps.com)

64. Navigate to your **Solution**

65. From the top menu select **Add existing > Process**

Currently the only way to create a business process flow directly in the solution is to use the classic solution explorer, for this course we have chosen to stay in the modern experience as much as possible. More details can be found here: [Microsoft Docs](#)

66. Select **Membership onboarding**

If you can't see it try refreshing the browser.

67. Click **Add**

The flow is now part of your solution!

---

## 9. Bringing the app together

### Publishing customisations

A number of customisations that we create first need to be published at the solution level before they are available for users to see and use. The following solution components require publishing when they are updated:

- Application Ribbon
- Entity\*
- Entity Relationship\*
- Field\*
- Form\*
- Message
- Option Set\*
- Site Map\*
- Web Resource

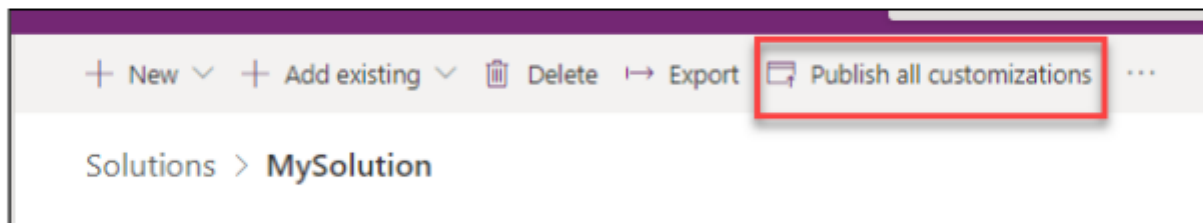
(We have created items with the asterisk\*)

---

#### Ex 9.1 - Publish all customisations

##### Continue from previous exercise

1. Within the solution click "Publish all customizations"



2. Our model-driven app is now ready to go!

---

### Sharing a model-driven app

Now that the model-driven app is ready. We can now share it with our audience. Sharing a model-driven app differs slightly from a Canvas App. We need to assign Dataverse roles when we share the app, compared to Canvas Apps these roles are much more flexible and granular. Where a standard role doesn't fit your needs, you can easily create a new role with customised permissions. More information about the predefined roles can be found here:

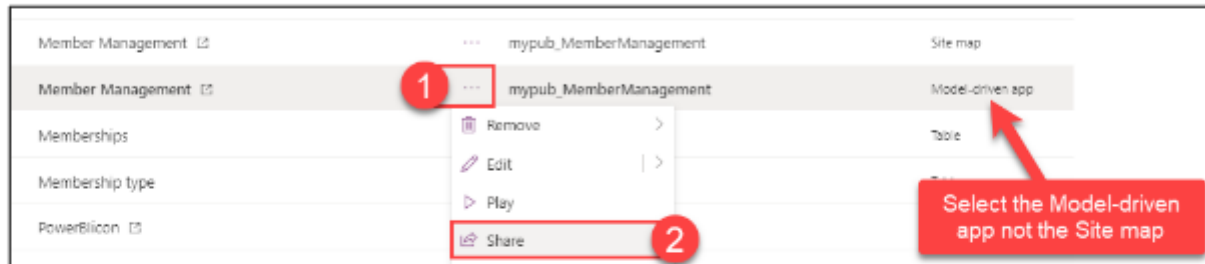
[Share a model-driven app using Power Apps - Power Apps | Microsoft Docs](#)



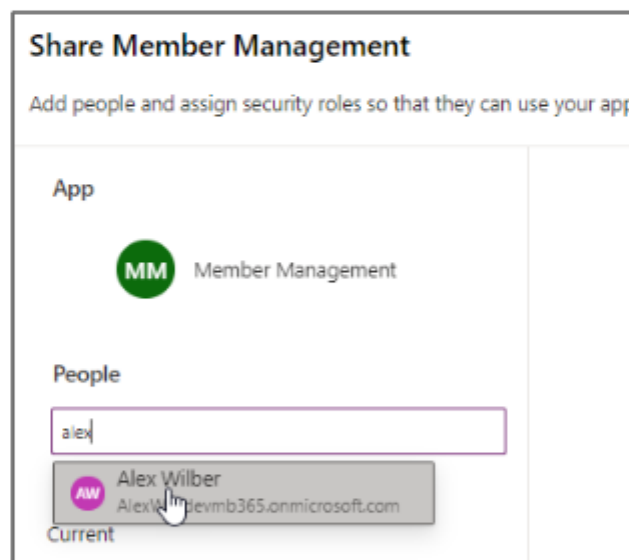
## Ex 9.2 - Sharing your completed app

### Continue from previous exercise

1. From within your Solution, click the **3-dots** of your Member Management Model-driven app
2. Select **Share**



3. Type in the email or name of a fellow student (or your instructor)  
Groups (like a Team) and Security groups can also be used to share model-driven apps.
4. From the dropdown box click the user to add the to the New section

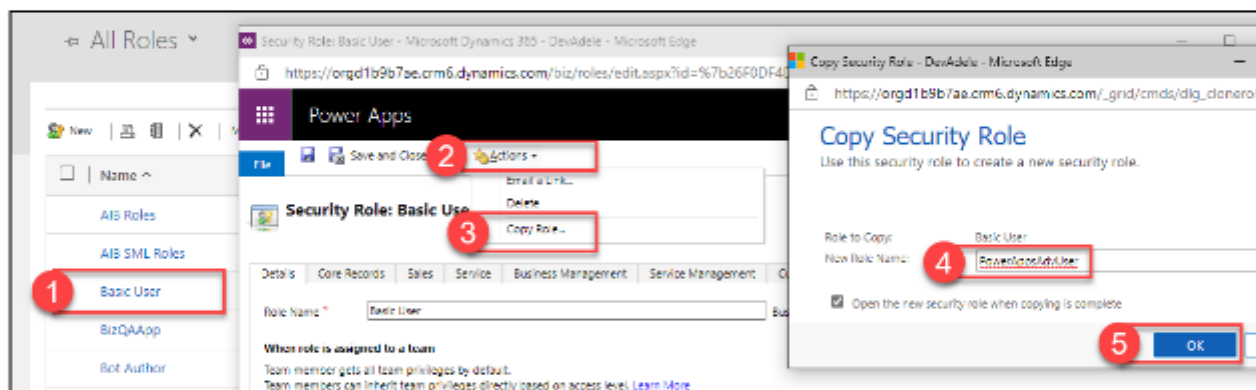


5. From the New section, click the **new user**
6. On the right-hand side of the menu, click the drop down menu next to Common Data Service/Dataverse, select **System customizer**
7. Click **Share**

## OPTIONAL EXERCISE

**Ex 9.3 - Modifying existing security permissions****Continue from previous exercise**

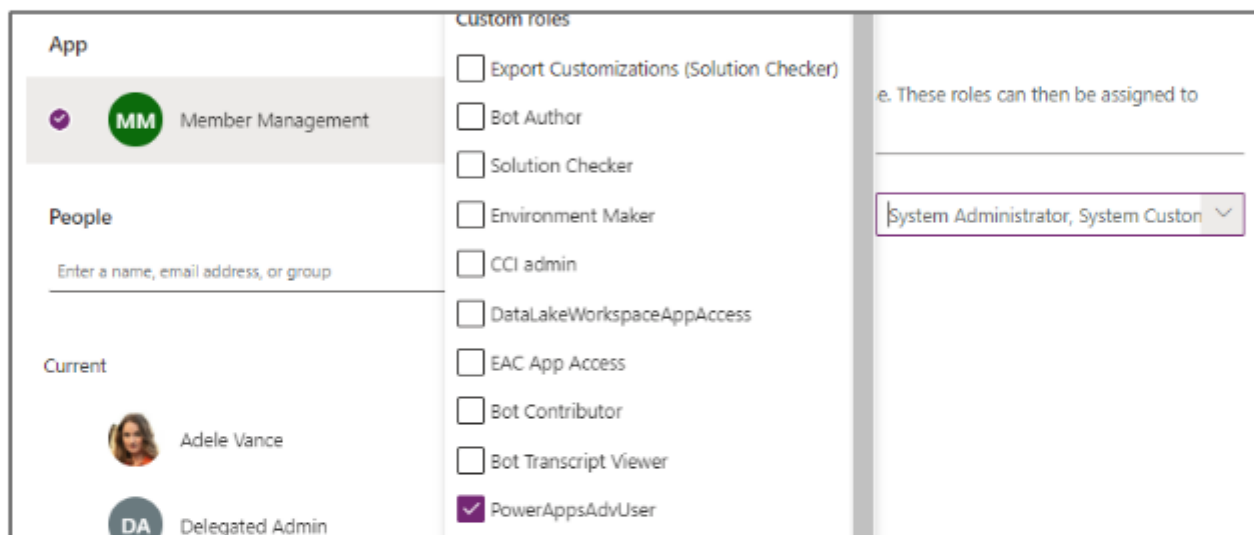
1. Repeat steps 1 to 5 from the above exercise
2. On the right-hand side of the menu, click the drop down menu next to Common Data Service/Dataverse, select **Manage Security Roles**  
(Note: Sometimes when a new window opens in Power Apps/Dynamics it looks like it is loading, but never does! Just close the window and try again, it usually works the second time)
3. Select the **Basic user** role
4. From the top menu click **Actions** and select **Copy Role...**
5. Name the new role **PowerAppsAdvUser**
6. Click **OK**



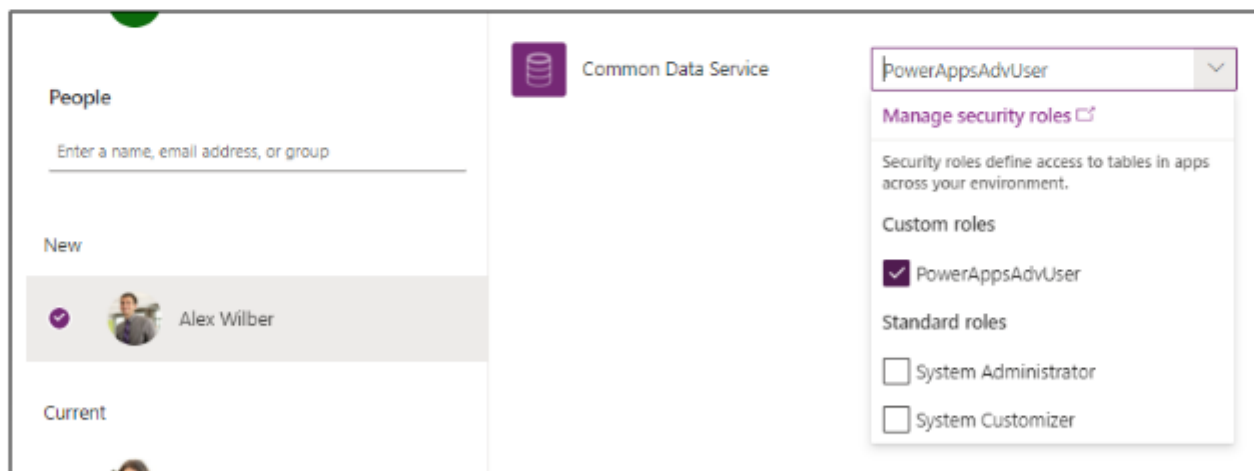
You may need to go back to the list of roles, refresh and select your new role

7. Click on the **Core Records** tab
8. Allow Organisation (full) rights on the **Contact** table by clicking each circle until it is fully green. (Click the table name to enable all circles)
9. Click on the **Business Process Flows** tab
10. Allow Organisation (full) rights on the **Membership onboarding** BPF by clicking the name of the flow.
11. Click on the **Custom Entities** tab
12. Allow Organisation (full) rights on the **Memberships** table by clicking the name of the flow.
13. Allow Organisation (full) **Read** rights on the **Membership type** table by clicking the second circle four times.

14. **Save and close** the new role
15. Return to the **Solution** screen
16. **Refresh** the browser to ensure the new roles are available
17. From within your Solution, click the **3-dots** of your Member Management Model-driven app
18. Select **Share**
19. Select the **Member Management App**
20. Click the drop down next to **Common Data Service** and select the role **PowerAppsAdvUser**



21. Add a new user and assign the new role to them



22. Click **Share**

---

Note that users don't currently receive an email when you share a model-driven app with them.

# 10. Power Automate

We have already used Power Automate to create standard, consistent business processes for our administration team to follow – called a business process flow. We can also create cloud flows to automate actions.

In this exercise we will trigger a flow when a new record is added to our Membership table, sending a welcome email to our new client.

Flows can be created as part of a solution. When it's part of a solution it's only visible through the solution. Like other Cloud flows it won't appear in your 'My flows' or 'Team flows' tabs.

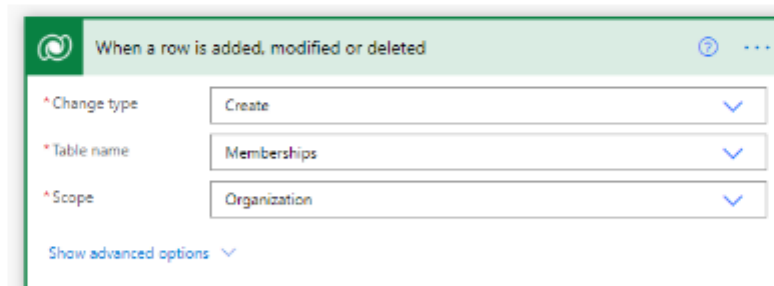
## Ex 10.1 - Creating a Cloud flow with the Dataverse

### Continue from previous exercise

1. From the top menu of your solution , click **New**, select **Cloud flow**

This will open a new window in Power Automate

2. Rename the flow from Untitled to **Welcome email flow**
3. In the trigger section, search for **Dataverse**
4. Select **When a row is added, modified or deleted**
5. Configure the following:

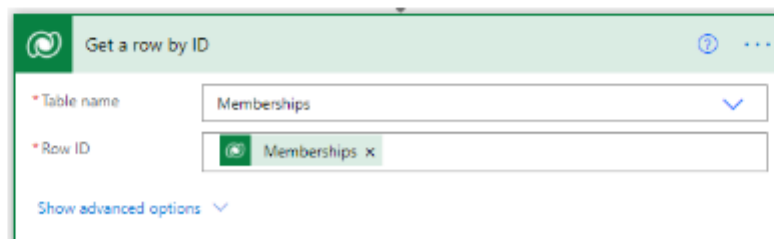


The screenshot shows the configuration for the trigger 'When a row is added, modified or deleted'. The fields are as follows:

| Field       | Value        |
|-------------|--------------|
| Change type | Create       |
| Table name  | Memberships  |
| Scope       | Organization |

There is a 'Show advanced options' link at the bottom.

6. Click **+ New step**
7. Select the Dataverse action **Get a row by ID** and configure:



The screenshot shows the configuration for the action 'Get a row by ID'. The fields are as follows:

| Field      | Value         |
|------------|---------------|
| Table name | Memberships   |
| Row ID     | Memberships x |

There is a 'Show advanced options' link at the bottom.

8. Click **+ New step**
9. Select the Dataverse action **Get a row by ID**

10. Click **Show advanced options** and configure:

Get a row by ID 2

\* Table name: Contacts

\* Row ID: ContactLU (Value)

Select columns: firstname

Expand Query: Enter an Odata style expand query to list related rows

Partition Id: An option to specify the partitionId while retrieving data for NoSQL tables

[Hide advanced options](#)

**Note:** The item in Row ID is ContactLU (Value)

11. Click + New step
12. Select the Office 365 Outlook action **Send an email (V2)**
13. Configure the action as follows using your accounts email address as a proxy for the customers:

Send an email (V2)

\* To: Adele Vance

\* Subject: Welcome to the Nexacu Gym

\* Body:
 

Font 12 B I U

Hi First Name

Thanks for joining the Nexacu Gym.

Soon you'll be receiving your Membership pack.

See you in the gym soon!

[Show advanced options](#)

14. Click **Show advanced options**
15. Set the Importance to **Normal**

Emails sent from Power Automate default to a Low Importance level.

16. Click **Save**

# 11. Putting the process together

We have now created:

- tables to store membership and member data,
- a canvas app for front-office staff to enter new member details
- a model-driven app for back-office staff to administer the onboarding process
- a business process flow to ensure a consistent onboarding experience
- a cloud flow to automate our welcome email to clients

We now have a workflow:

1. New member details are entered into the Gym Membership app.
2. An email is sent to your email for each record you create.
3. In the Member Management app, new records will appear in the 'Membership packs not sent' view.
4. Clicking on a record will open it up in the form, with the business process flow shown across the top.

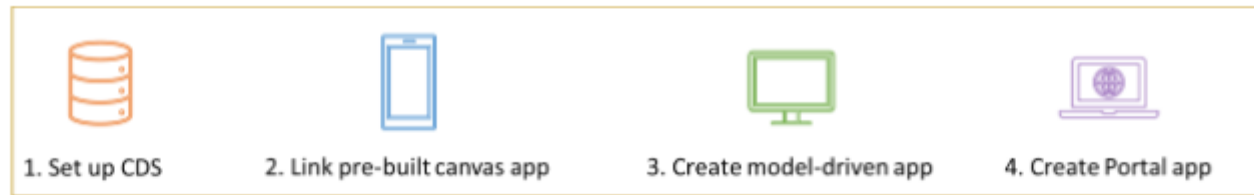
Depending on the membership level, 2 to 4 stages will appear to guide admin staff through the onboarding process.

**Note:** If you click on a record that was created prior to the business process flow being created, it won't have the process assigned to it.

You can assign it by opening the record and clicking Process > Switch Process > Member onboarding > OK



## 12. Portal



Model-driven and canvas Power Apps deliver business functionality to people *within* an organisation and with appropriately licensed guests from other organisations. Power Apps **portals** extend access to the Dataverse to internal and external audiences such as communities, customers, partners, and employees.

To access a portal, users don't need to ensure that they are licensed as this is provided by the hosting organisation. Licensing is based upon the number of pages views and if the user is viewing the portal as an authenticated user or anonymously.

Portals aren't provisioned automatically when a new environment is created and only one portal per language can be created in an environment. To provision a power apps portal you must be assigned the System Administrator role of the environment. When the portal is being provisioned it creates a number of supporting **tables**, as well as a **portal management app** (model-driven) and the actual **Portal app**.

Portals can't be added to a solution like canvas and model-driven apps, instead you can use 3<sup>rd</sup> party tools to import and export your portals. More info [here](#).

In chapter 6, we started the portal app provisioning process – this should now be completed and available in the environment.

| Apps    Component libraries (preview) |                   |            |             |              |
|---------------------------------------|-------------------|------------|-------------|--------------|
|                                       | Name              | Modified   | Owner       | Type         |
|                                       | Gym Portal        | 11 min ago | Adele Vance | Portal       |
|                                       | Portal Management | 1 min ago  | Adele Vance | Model-driven |
|                                       | Gym membership    | 1 wk ago   | Adele Vance | Canvas       |
|                                       | Member Management | 1 wk ago   | Adele Vance | Model-driven |

The portal (i.e. the website) is stored in Dataverse tables. Everything from page templates to content is stored in different tables.

The Portal Management app allows us to configure the app through a, now, familiar model-driven app experience. We can also use the Portal designer to change the app in a *What You See Is What You Get* (WYSIWYG) experience.

While we can edit content in the portal designer. We need to use the Portal management app for some configurations such as security.



**Ex 12.1 - Viewing the portal app**

1. Go to <https://make.powerapps.com/>
2. Select your environment
3. Click the **Gym Portal** app to open it in a new browser  
Notice the URL – yep you've created your own website!
4. Explore the pre-built website

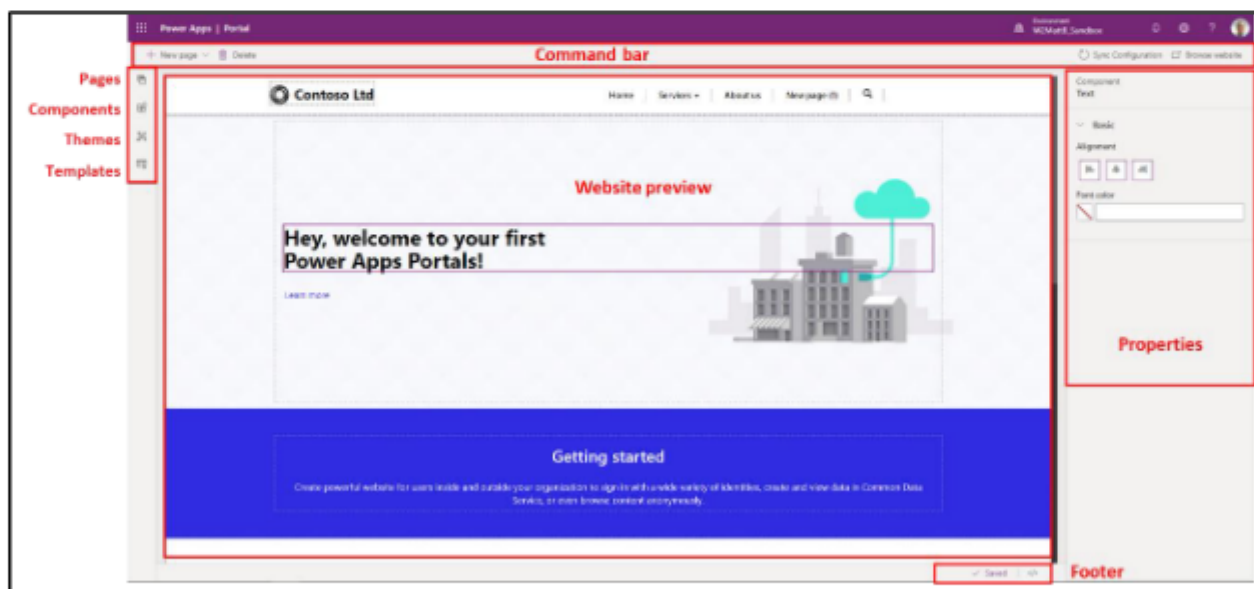
**Exploring the portal designer**

The portal designer allows users to create their website in an easy to use What You See Is What You Get experience. Updating content, resizing images and adding pages can all be done using the portal designer.

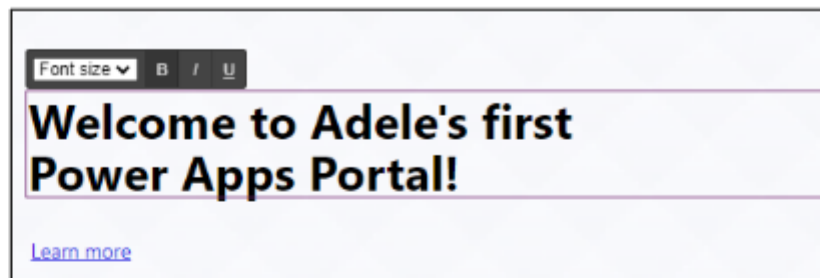
**Ex 12.2 - Changing content in the Portal designer****Continue from the previous exercise**

1. Go to <https://make.powerapps.com/>
2. Select your environment
3. Click the 3-dots of the **Gym Portal** app and select **Edit**

This opens the portal designer:



4. Change the home page's text from *Hey, welcome to your first Power Apps Portals!* to: **Welcome to [Your Name]'s first Power Apps Portal!**



5. In the top right corner select **Browse website**

This will save the changes and open the website in a new browser tab.

6. Return to the portal designer

7. On the left menu click the **Pages** icon

8. Select the **About us** page

9. Click on the image of the people on the bicycle

This will open the properties of the image on the right-hand side. But we can also change the underlying code, in this case CSS, in the portal designer.

10. In the bottom right-hand footer, click the **code** icon:



This opens the code for the selected component. Portal apps are written using a combination of [HTML](#), [CSS](#) and Liquid. Liquid is an open-source template language integrated into portals. It can be used to add dynamic content to pages, and to create a wide variety of custom templates.

More information about Liquid can be found here: [Work with Liquid template language in Power Apps portals - Learn | Microsoft Docs](#)

11. In the third row of code you will find the **IMG** declaration

12. Change the image's width property to **70%**



## Exploring the Portal Management app

The Portal Management App is a model-driven app that allows us to configure the Portal's data stored in a number of Dataverse tables. Through this app we are able to change all settings of the Portal including pages, content and security.

### Ex 12.3 - Configuring the portal through the management app

1. In a new browser tab, go to <https://make.powerapps.com/>
2. Select your environment
3. Open the **Portal Management App**
4. On the left menu, under *Content*, click **Web Pages**
5. Select the **About Us** web page
6. Under *Localized Content*, select **About Us**
7. Scroll down to the *Content* section and click the **HTML** tab
8. Locate the code that was changed in the previous exercise using the Portal designer

As you can see, any changes made in the Portal designer are reflected in the Portal Management App.

If you make any changes in the Portal Management App while you have the portal designer open you will need to sync the changes to see them in the portal designer. In the top right corner of the portal designer just click **Sync configuration**.

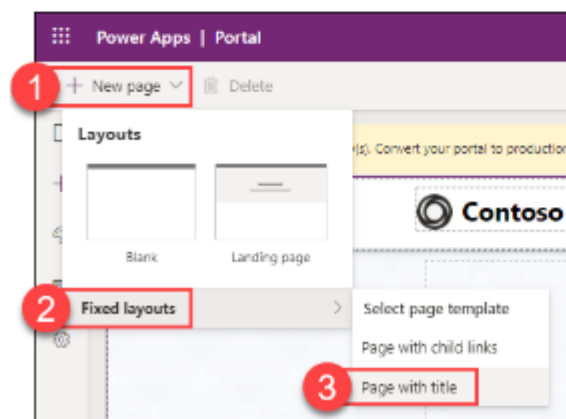
## Adding and moving portal pages

With the portal designer it is very easy to add new pages to the site and configure the hierarchy of pages.

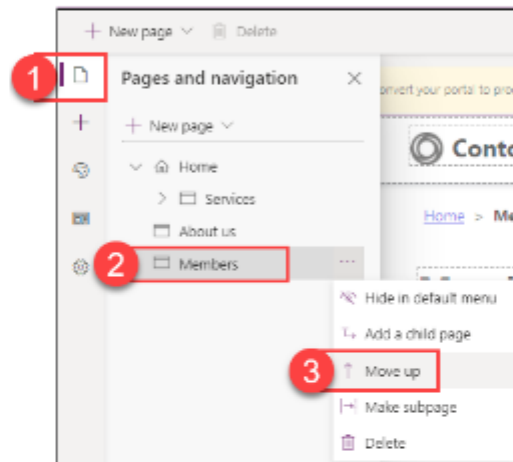
### Ex 12.4 - Creating pages in the portal

#### Continue from the previous exercise

1. Return to the **portal designer**
2. In the top left corner, click **+ New page**
3. Click **Fixed layouts**
4. Select **Page with title**



5. Change the new page's title to **Members**
6. In the page's properties also change the Name and Partial URL to **Members**
7. Open **Pages and Navigation** in the left rail menu
8. Click the **3-dots** next to the Members page
9. Move the page up once
10. Click the **3-dots** next to the Members page again
11. Click **Make subpage** of Services



---

## Displaying Dataverse data on a page

The real benefit of the portal app is being able to access and display data from our Dataverse tables in an external facing website.

In the next exercise we will display our membership details on the Member's page.

---

### Ex 12.5 - Displaying Dataverse data on a page

#### Continue from the previous exercise

1. In a new browser go to <https://make.powerapps.com/>  
Leave the portal designer tab open as we'll be coming back to it
2. Select your **environment**
3. Open your **solution**  
(You'll notice many new solutions now exist due to the portal app)
4. Select the **Membership** table
5. Click on the **Views** tab

6. Add a **New view** called **Webview**
7. Along with Member Number, add:
  - Full name (related column from ContactLU)
  - Join Date
  - MemberType (related column from MembershipTypeLU)

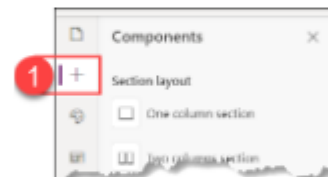
| Member Number | Full Name ... | Join date | MemberTy... | + View column |
|---------------|---------------|-----------|-------------|---------------|
| GYM-1000      | Tom Cat       | 6/22/2021 | Silver      |               |
| GYM-1001      | Jerry Mouse   | 6/22/2021 | Gold        |               |
| GYM-1002      | Mickey Mouse  | 7/15/2021 | Platinum    |               |
| GYM-1003      | Donald Duck   | 7/15/2021 | Platinum    |               |

For this exercise we'll skip assigning filters and sorting rules

8. **Save and Publish** the view
9. Return to the Solution and **Publish all customizations**
10. Navigate back to the **portal designer** and refresh the page
11. In the top right corner click on **Sync configuration**

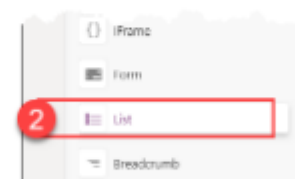
This is just to make sure that our portal designer has all the latest connections to our data

12. From the left rail menu click on the **components** icon (plus icon)



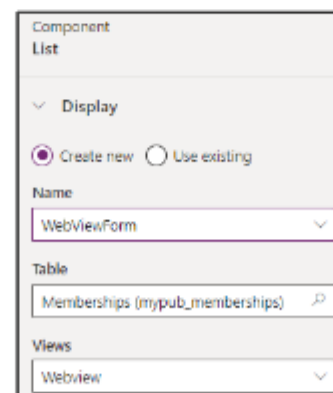
13. Select the **List** component

With the list component selected its properties pan will display on the right



14. Select **Create new**
15. Name the list: **WebViewForm**
16. Select the table: **Memberships**
17. Select the view: **Webview**
18. In the top right corner click **Browse website**

The data should now be visible. We have noticed a bug where you might need to first log into the website (using the Sign In > External SignIn > Azure AD option) before records are shown. You can then sign out and the records should still be visible.



## Restricting who can view a page

Often, we want to restrict who can view a certain page on a site.

Let's say that we want to restrict our membership page, containing member data, to only our administrators.

To do this we first create a **Web page access control rule**

We'll then assign an existing **Web Role** called Administrators to the access control rule.

Security configurations need to be setup via the Portal Management App.

One thing to take note of, in the portal designer you are always signed in as an administrator. This allows you to view all pages, lists, etc.

---

### Ex 12.6 - Creating a Web Page Access control rule

#### Continue from the previous exercise

1. Navigate to your environment's **Portal Management** app
2. On the left hand menu, under Security, click **Web Page Access Control Rules**
3. At the top of the page click **+ New**
4. Name the new rule: **Members Only**

Website: **Starter Portal**

Web Page: **Members**

Right: **Restrict Read**

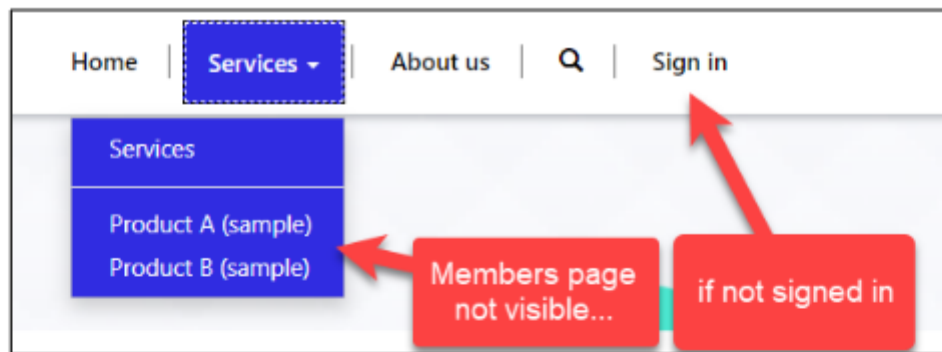
When you create a restrict rule for a page, the page is blocked for all. And then you assign webroles to this rule to explicitly make this page available to users in that webrole.

5. Click **Save**

If you don't click save the next options won't appear.

6. Click the **Web Roles** tab
7. On the right side of the web roles tab, click **Add Existing Web Role**
8. Search for and select **Administrators**
9. Click **Add**
10. Click **Save**
11. **Refresh** the portal website

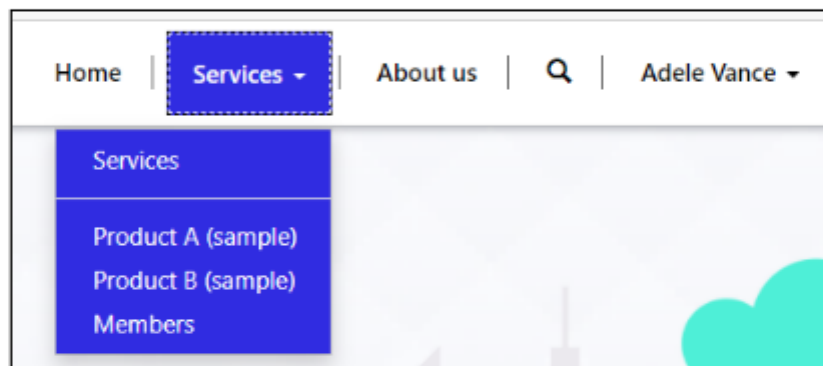




12. Click **Sign in**

13. Click **Azure AD**

Accept any messages to continue.



The page should now be visible