Microsoft

PowerApps Bootcamp: Advanced

Hands-On Lab Guide

Published (Jan 2025)

Microsoft
Power Platform

Microsoft
Power Platform

Disclaimer

Microsoft Power Platform

Contents

Microsoft Power Platform

Case Study: Equipment Monitor

Scenario

You are part of the **Champions Team**, a group of motivated individuals who actively identify inefficient processes, rethink how things are done, and introduce smarter solutions using Microsoft Power Platform.

One of the challenges your team is tackling involves managing office equipment, which is currently tracked through a mix of **Telegram messages, scattered Excel files, and emails**. This fragmented approach has led to confusion, duplication, and missing records.

To address this, your team is leading the initiative to revolutionize equipment tracking by building a centralized **Inventory Management App** using Power Platform.

Goals of the Solution

The app must:

- Maintain a real-time **inventory list** of all office equipment
- Allow users to **check items in and out** easily
- Log **service or repair history**
- **Display item Inventory Status and updates** in real time for all users

You will design and build:

1. A structured data model to support Items, Users, Check In/Out records, and Service Logs
2. A multi-screen Canvas App with a modern, responsive layout
3. Automated logic to validate and manage checkouts
4. Optional flows for notifications and reminders

Commented [GP1]: Before Exercise 1 explore possibility to run through slides to share the various data schems, data structures, delete methods etc.

With the scenario let them identify user story As a ____, I would like to _____ so that I can ______.
Defining the features etc
Basic Time motion study

Microsoft Power Platform

Exercise 1: Understanding and Building the Data Schema

As part of the *Equipment Monitor* initiative, you will define the foundational data model for a centralized inventory management app. This model supports tracking of items, users, service activity, and item check-in/check-out. In this exercise, you will review the schema, understand the relationships, and build the tables using **Microsoft Dataverse**.

Objectives

After completing this exercise, participants will be able to:

- Understand the core entities and their relationships in the Equipment Monitor app
- Identify appropriate Dataverse data types (e.g. text, choice, lookup, AutoNumber)
- Create normalized tables and relationships in Dataverse

Requires

- “Sample_Data_Tables.xlsx”

Estimated time to complete this lab

60 mins

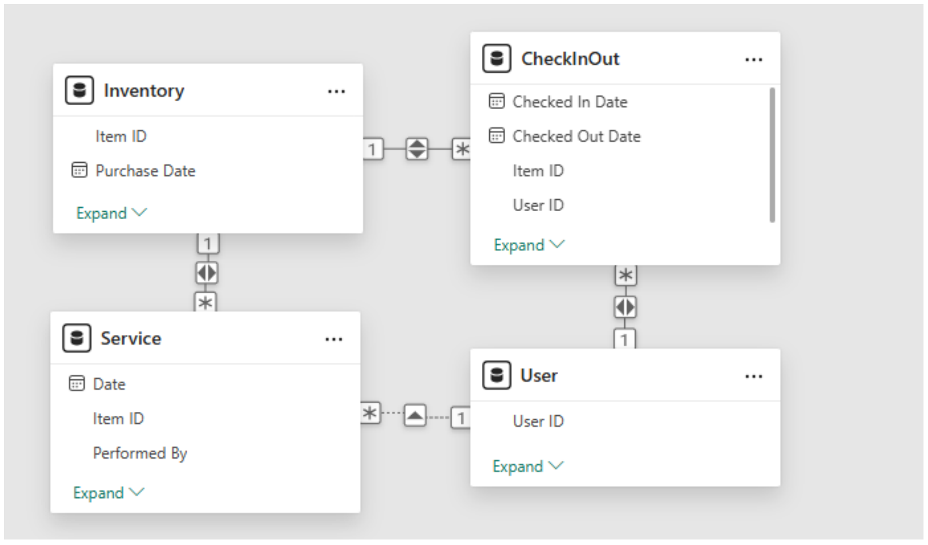
Task 1: Understand the Data Model

Understanding Relationship Types in Dataverse
Dataverse supports three primary types of data relationships:

- **One-to-Many (1:N):** A single record in one table relates to multiple records in another. Example: One **Inventory** item may have many **Service** records.
- **Many-to-One (N:1):** Many records relate back to a single parent record. Example: Multiple **CheckInOut** entries point to the same **User**.
- **Many-to-Many (N:N):** Records in both tables can relate to multiple records in the other. While Dataverse supports N:N relationships natively, in this case, we simulate it using the **CheckInOut** table as a junction between **Inventory** and **User** — allowing each item to be checked out multiple times by multiple users, while storing transaction details.

The Equipment Monitor app is structured around four core data tables:

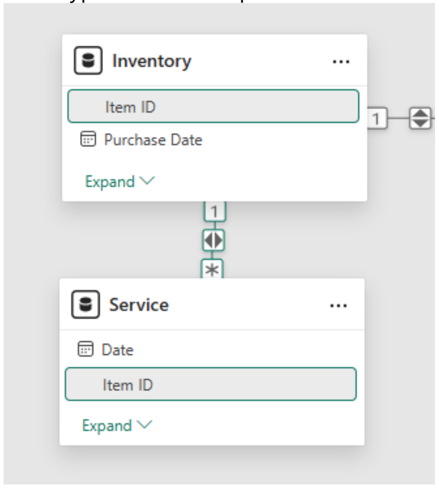
Table	Description
Inventory	Holds details of equipment items available for use or loan
User	Contains employee information — users may check out items or perform servicing
Service	Records maintenance or repair activity for individual inventory items
CheckInOut	Tracks when users borrow or return equipment, including condition and timestamps



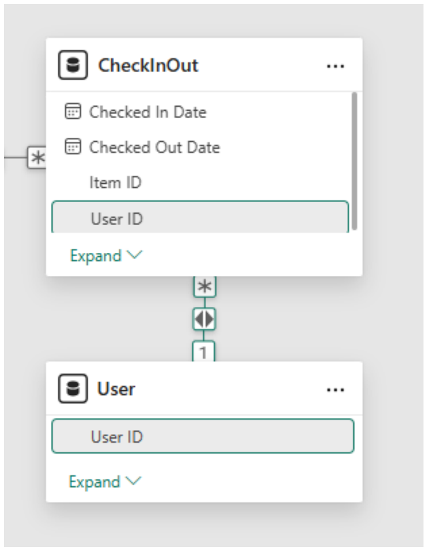
Microsoft Power Platform

Instructions:

1. Based on the description and schema:
   - What type of relationship exists between **Inventory** and **Service**?



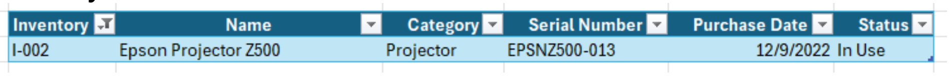
   - What is the nature of the link between **CheckInOut** and **User**?



2. **Understand Referential Behaviour in Relationships**
   In Dataverse, when creating relationships between tables, you can define how records behave when their parent is deleted. This is called **referential or cascade behaviour**. There are several types:
   - **Parental (Cascade All):** When a parent record is deleted, all related child records are also deleted automatically.
     Example: If an Inventory item is deleted, all its Service records are also deleted.
   - **Restrict Delete:** Prevents the deletion of a parent record if child records exist.
     Example: You cannot delete a User if they have CheckInOut records unless those are removed first.
   - **Cascade Active:** Only active (not inactive or archived) child records are deleted when the parent is deleted.
   - **Set Null:** When a parent record is deleted, the child record's lookup field is set to blank.
   - **Remove Link:** The relationship between parent and child is removed, but both records remain.

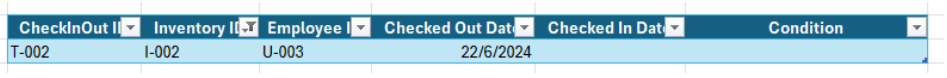
Best Practice: Use *Restrict Delete* for transactional data (like CheckInOut), and *Parental* only when child records have no relevance without their parent.

Inventory Table



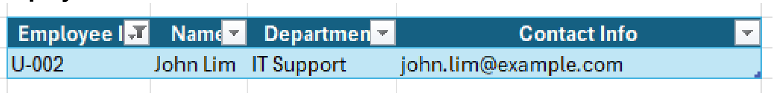
Inventory	Name	Category	Serial Number	Purchase Date	Status
I-002	Epson Projector Z500	Projector	EPSNZ500-013	12/9/2022	In Use

CheckInOut Table



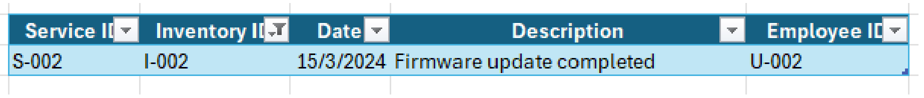
CheckInOut I	Inventory I	Employee I	Checked Out Dat	Checked In Dat	Condition
T-002	I-002	U-003	22/6/2024		

Employee Table



Employee I	Name	Departmen	Contact Info
U-002	John Lim	IT Support	john.lim@example.com

Service Table



Service I	Inventory I	Date	Description	Employee I
S-002	I-002	15/3/2024	Firmware update completed	U-002

From the above tables we understand that the Epson Projector Z500 is currently still on Loan to Sarah and has been serviced once on 15 March 2024

--- **End of Task** ---

Task 2: Identify and Map Data Types

Use the following tables to guide your setup in Dataverse. The columns now include whether a field should have a default value and if it should be made required.

Understanding Dataverse Data Types

Dataverse provides various data types to help structure your tables effectively. Below are the most common types, grouped by their purpose:

1. **Text and Descriptive Fields**
   - Text: For short strings like names, titles, and labels
   - Multiline Text: For longer descriptions, comments, or notes
   - Email: For validated email addresses
   - Phone: For phone numbers with formatting support
   - URL: For website links (optional use)
2. **Choice-Based Fields**
   - Choice: Select one value from a predefined list (e.g., Category: Laptop, Monitor, etc.)
   - Choices: Select multiple values from a list (multi-select)
   - Two Options: For Yes/No or True/False decisions (e.g., Is Returned)
3. **Numbers and Financial Fields**
   - Whole Number: For integers like quantity or count
   - Decimal Number: For values with decimal precision
   - Currency: For monetary amounts, supports formatting and precision
4. **Date and Time Fields**
   - Date and Time: Used to capture a calendar date or timestamp (e.g., Purchase Date, Return Date)
5. **Relational Fields**
   - Lookup: Links to a record in another table, used to define relationships (e.g., User, Inventory)
   - Customer: A special type of lookup used mostly in model-driven apps to point to either Accounts or Contacts (not needed here)
   - Owner: System-managed, identifies who owns a record (used in Dataverse security model)
6. **System and Utility Fields**
   - Autonumber: Automatically generates unique values for primary fields (e.g., INV-001, SRV-001)
   - Calculated: Derived values based on expressions (used rarely at this level)
   - Rollup: Aggregates data from related records (advanced usage)

7. **File and Media (Optional)**
   - File: Allows file attachments
   - Image: Stores one image per record (e.g., for displaying item photos in a gallery)

Inventory Table

Field Name	Description	Data Type	Default Value?	Required?
Inventory ID	Unique Inventory IDentifier	Autonumber (Primary)	Yes (System)	Yes
Name	Item name or label	Text	No	Yes
Category	Type of item (e.g. Laptop, Monitor)	Choice	No	Yes
Serial Number	Manufacturer serial number	Text	No	Yes
Purchase Date	Date item was acquired	Date and Time	No	Optional
Inventory Status	Availability (Available/In Use/Under Repair)	Choice	Yes ("Available")	Yes
Notes	Additional details or comments	Multiline Text	No	Optional

Employee Table

Field Name	Description	Data Type	Default Value?	Required?
Employee ID	Unique identifier	Autonumber (Primary)	Yes (System)	Yes
Full Name	Name of the user	Text	No	Yes
Department	Department user belongs to	Choice	No	Optional
Email	Contact email	Email	No	Yes
Phone Number	Optional mobile number	Phone	No	Optional

Service Table

Field Name	Description	Data Type	Default Value?	Required?
Service ID	Unique service entry	Autonumber (Primary)	Yes (System)	Yes
Inventory	Related item	Lookup (Inventory)	No	Yes
Employee ID	User who performed the service	Lookup (User)	No	Yes
Service Date	Date service was done	Date and Time	No	Yes
Description	What was serviced or repaired	Multiline Text	No	Yes
Cost	Optional cost of service	Currency	No	Optional

CheckInOut Table

Field Name	Description	Data Type	Default Value?	Required?
Transaction ID	Unique transaction record	Autonumber (Primary)	Yes (System)	Yes
Inventory	Item being checked out/in	Lookup (Inventory)	No	Yes
User	Person performing check-out/in	Lookup (User)	No	Yes

Checked Out Date	When the item was borrowed	Date and Time	No	Yes
Checked In Date	When the item was returned	Date and Time	No	Optional
Condition on Return	State of the item upon return	Multiline Text	No	Optional
Is Returned	Whether the item has been returned	Two Options (Yes/No)	Yes ("No")	Yes

Tip : When in doubt, prefer the following

Choice over free text for controlled vocabularies

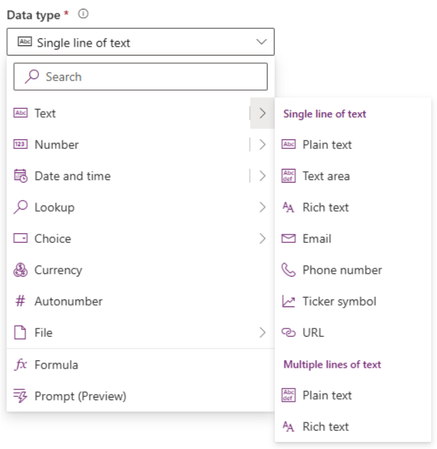
Lookup for any field that references another table

Two Options for binary logic (true, false)

Autonumber for consistent, non-manual IDs

Instructions:

1. Use the table above as your reference for creating columns in Dataverse.
2. Consider where *Choice*, *Lookup*, or *Yes/No* fields are best suited instead of free-text to enforce data integrity.
3. Think ahead: Which fields should be required? Which should allow multiple lines? What should be searchable?



-- **End of Task** --

Microsoft Power Platform

Task 3: Creating Solution and Importing the Tables in Dataverse

In this task, you will build the four required tables for the Equipment Monitor solution using Dataverse. You will configure the appropriate relationships and data types based on the schema discussed earlier.

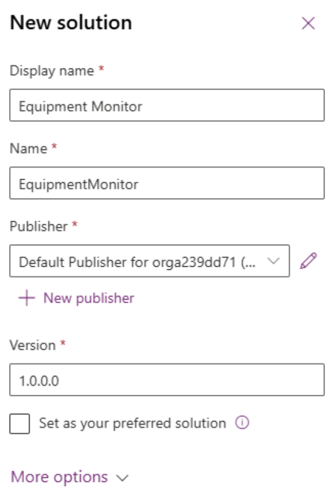
1. Go to Power Apps Maker Portal

2. In the left-hand panel, click on **Solutions > + New solutions**

3. Create **New publisher**

   Display name: **Your Display Name**
   Name: **Name w/o Spacing ' '**
   Prefix: **Your Initials**

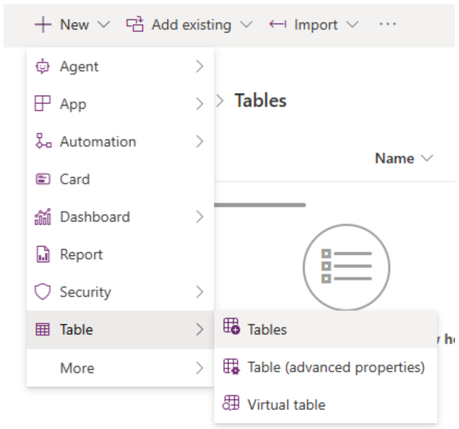
4. Create **New solution**



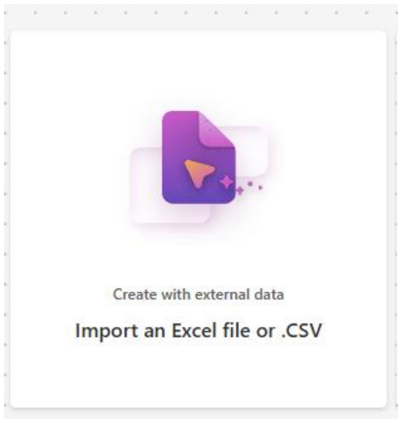
New solution

Display name *	Equipment Monitor
Name *	EquipmentMonitor
Publisher *	Default Publisher for orga239dd71 (...
Version *	1.0.0.0

5. Click **+ New > Table > Tables**

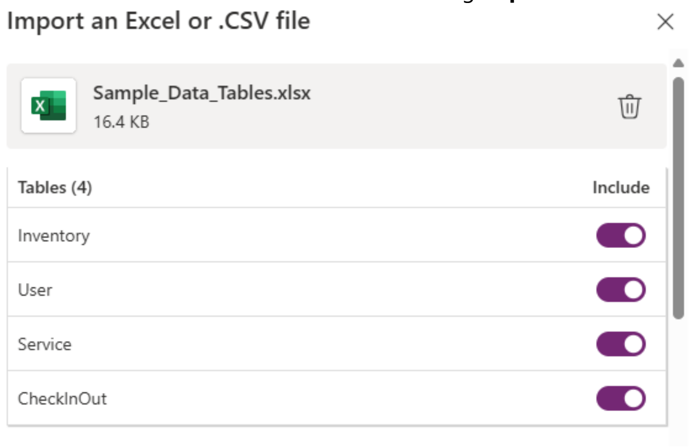


6. Select **Import an Excel file or .csv**



7. Select **'Sample_Data_Tables.xlsx'** file

8. Ensure that all tables are included before selecting **Import**.



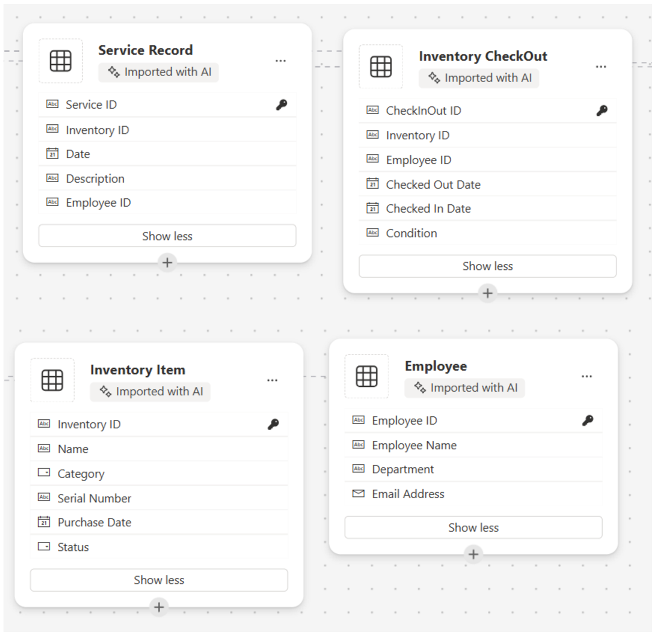
Import an Excel or .CSV file

Sample_Data_Tables.xlsx
16.4 KB

Tables (4)	Include
Inventory	●
User	●
Service	●
CheckInOut	●

9. Ensure the following tables are created **Inventory Item**, **Employee**, **Service Record**, **CheckInOut Record**

10. Set following as Primary Columns **Inventory ID**(Inventory Item), **Employee ID**(Employee), **CheckInOut ID**(CheckInOut Record), **Service ID**(Service Record)



Service Record
Imported with AI
- Service ID
- Inventory ID
- Date
- Description
- Employee ID

Show less

Inventory CheckOut
Imported with AI
- CheckInOut ID
- Inventory ID
- Employee ID
- Checked Out Date
- Checked In Date
- Condition

Show less

Inventory Item
Imported with AI
- Inventory ID
- Name
- Category
- Serial Number
- Purchase Date
- Status

Show less

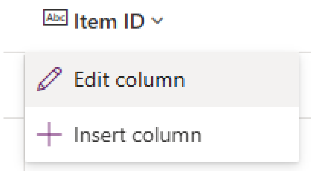
Employee
Imported with AI
- Employee ID
- Employee Name
- Department
- Email Address

Show less

Microsoft
Power Platform

11. Select Card and More settings > **View data**

12. We will ensure that all the data columns are in the correct **Data type**.



<u>Inventory Item Table</u>
Inventory ID: **Single line of text - Text**
Name: **Single line of text - Text**
Category: **Choice**
Serial Number: **Single line of text - Text**
Purchase Date: **Date and time – Date only**
Inventory Status: **Choice**

<u>Employee Table</u>
Employee ID: **Single line of text - Text**
Full Name: **Single line of text - Text**
Department: **Choice**
Email Address: **Single line of text – Email**

<u>Service Record Table</u>
Service ID: **Single line of text - Text**
Inventory ID1: **Single line of text - Text**
Date: **Date and time – Date only**
Description: **Single line of text - Text**
Employee ID1: **Single line of text – Text**

<u>CheckInOut Record Table</u>
CheckInOut ID: **Single line of text - Text**
Inventory ID1: **Single line of text - Text**
Employee ID1: **Single line of text - Text**
Checked Out Date: **Date and time – Date only**
Checked In Date: **Date and time – Date only**
Condition: **Single line of text - Text**

--- **End of Task** ---

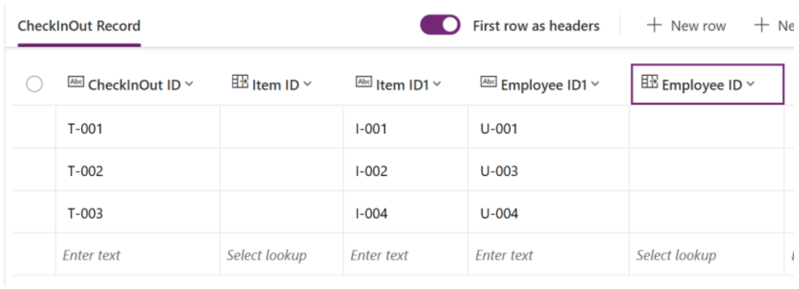
Microsoft Power Platform

Task 4: Creating Dataverse Relationship

1. For **Service Record** Table and **CheckInOut Record** Table
   Insert the following columns

   Service Record Table
   - Inventory ID – Lookup – Inventory Item
   - Employee ID – Lookup – Employee

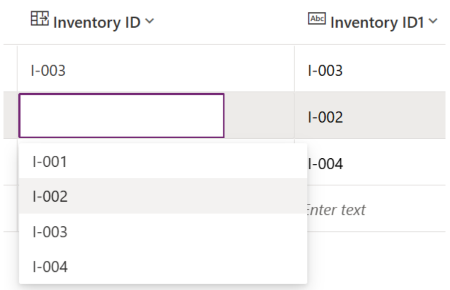
   CheckInOut Record Table
   - Inventory ID – Lookup – Inventory Item
   - Employee ID – Lookup – Employee



CheckInOut Record | First row as headers | New row | Ne

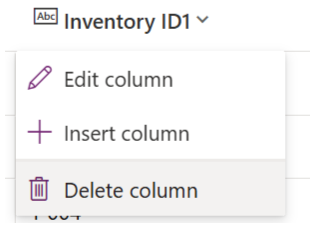
CheckInOut ID	Item ID	Item ID1	Employee ID1	Employee ID
T-001		I-001	U-001	
T-002		I-002	U-003	
T-003		I-004	U-004	
Enter text	*Select lookup*	*Enter text*	*Enter text*	*Select lookup*

2. Based on the Text Column, we will now populate the Lookup Columns we have created



Inventory ID	Inventory ID1
I-003	I-003
	I-002
	I-004
	Enter text

3. We will then delete the columns we have renamed Inventory ID1 and Employee ID1



4. Click on **Save and exit** to import the fields.

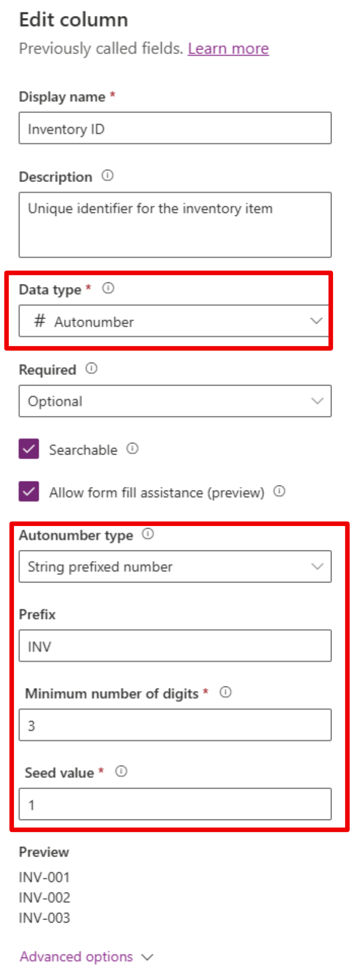
-- **End of Task** --

Microsoft
Power Platform

Task 5: Configuring Auto Numbering Column

1. On the Left Pane **Table>Columns>{Column}**
   Change the primary column to use **Autonumber**, and set a prefix such as: INV-{001}

   Inventory ID (Inventory Item) – INV-001
   Employee ID (Employee) – EMP-001
   CheckInOut ID (CheckInOut Record) – TRA-001
   Service ID (Service Record) – SVC-001



Edit column
Previously called fields. Learn more

Display name *
Inventory ID

Description
Unique identifier for the inventory item

Data type *
Autonumber

Required
Optional

✓ Searchable

✓ Allow form fill assistance (preview)

Autonumber type
String prefixed number

Prefix
INV

Minimum number of digits *
3

Seed value *
1

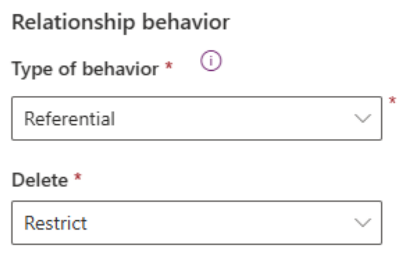
Preview
INV-001
INV-002
INV-003

Advanced options

2. On the Left Pane **Table>Relationship**
   For each lookup, configure the **relationship behaviour**. Use **Referential, Restrict** to prevent deletion of parent records that have related child records.

   Service Record ⇔ Employee
   Service Record ⇔ Inventory Item

   CheckInOut Record ⇔ Employee
   CheckInOut Record ⇔ Inventory Item


Relationship behavior

Type of behavior *: Referential

Delete *: Restrict

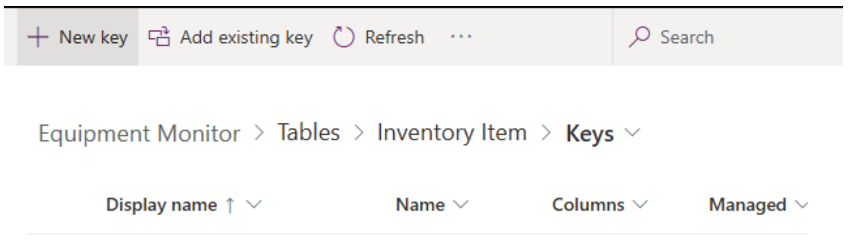
3. Click **Save**

--- **End of Task** ---

Task 6: Configure Alternate Keys (Optional)

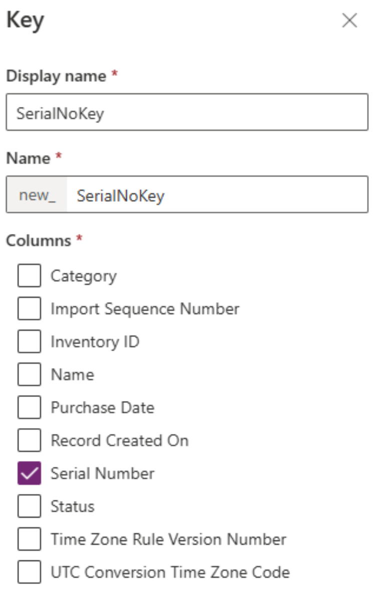
This task is optional but recommended if your app requires preventing duplicate records — especially for fields like serial numbers or user emails.

1. Open the **Inventory Item** table
2. Click on the **Keys** tab



4. Click **+ New Key**
5. Select the **Serial Number** column to create a key
   - This will ensure each serial number in the Inventory table is unique

- You can also repeat this for the **Email** column in the **User** table if needed



Key

Display name *
SerialNoKey

Name *
new_ SerialNoKey

Columns *

- [ ] Category
- [ ] Import Sequence Number
- [ ] Inventory ID
- [ ] Name
- [ ] Purchase Date
- [ ] Record Created On
- [x] Serial Number
- [ ] Status
- [ ] Time Zone Rule Version Number
- [ ] UTC Conversion Time Zone Code

6. Click **Save**

--- **End of Task** ---

Microsoft Power Platform

Exercise 2: Create a Model-Driven App to Manage Inventory and Users

In this exercise, you will create a model-driven app to help Power Platform champions or administrators manage backend records such as inventory items and users. Model-driven apps offer a fast, secure, and scalable way to manage Dataverse tables using automatically generated forms and views.

Objectives

After completing this exercise, participants will be able to:

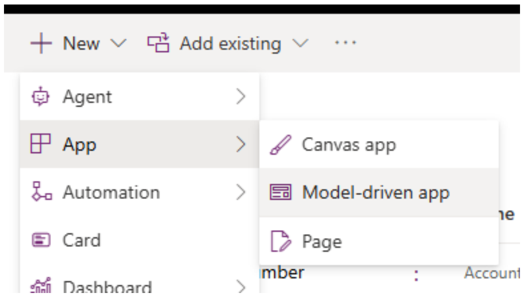
- Create a model-driven app using Dataverse tables
- Add Inventory and User tables to the app navigation
- Understand and customize forms and views
- Test and use the app for backend data management

Estimated Time

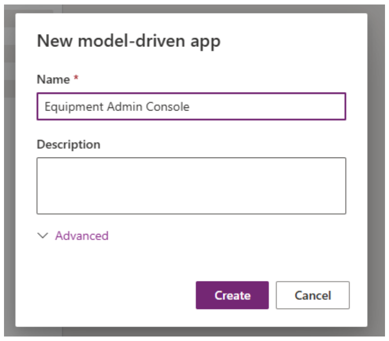
30–40 minutes

Task 1: Create a Model-Driven App

1. Go to Power Apps Maker Portal
2. In the left menu, click **Apps**
3. Click **+ New app** > **Model-driven app**



4. Name the app: Equipment Admin Console

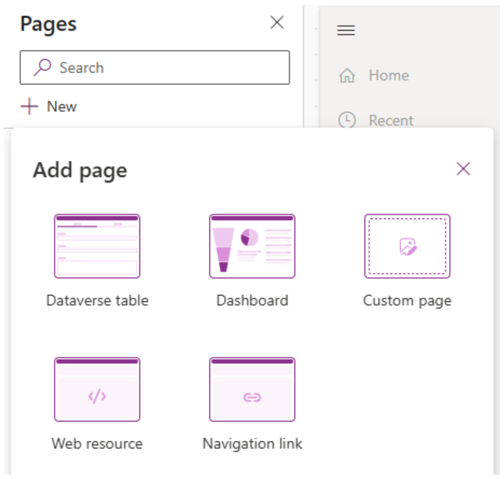


5. Click **Create**

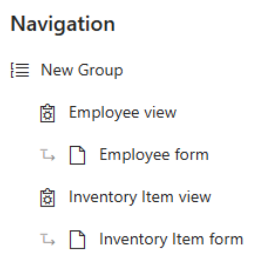
-- End of Task --

Task 2: Add Tables to the App Navigation

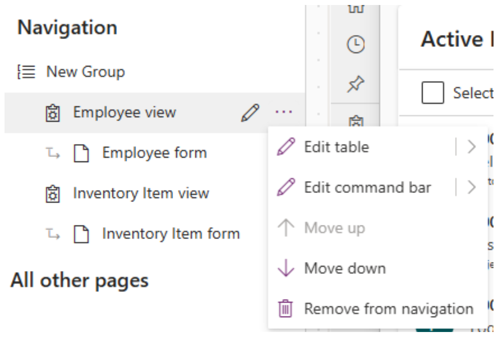
1. In the **App Designer** canvas, click **+ Add Page**
2. Select **Dataverse table**, then click **Next**



3. Choose the **Inventory Item** table and **Employee** table, then click **Add**



5. Reorder and rename the navigation if needed using drag-and-drop



7. Click **Save**, then **Publish**

-- End of Task --

Understanding Views and Forms in Model-Driven Apps

Views

- **Public Views**: Shared with all users
- **Personal Views**: Private, user-defined filters and layouts
- Views can be edited to show or hide columns, filter records, and sort data

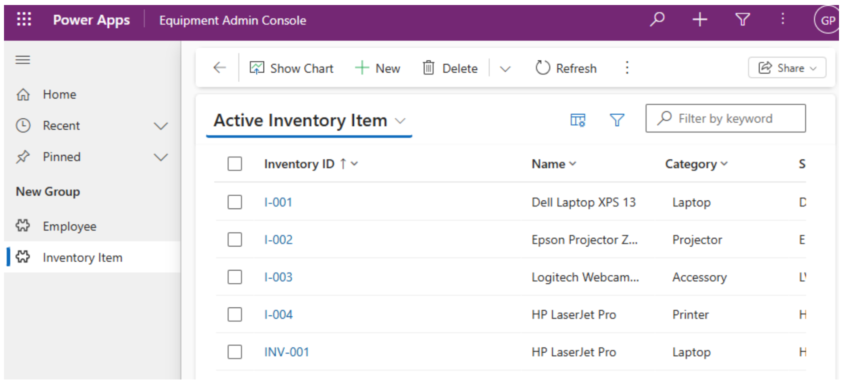
Forms

- **Main Forms**: Full layout for viewing/editing a record
- **Quick Create Forms**: Lightweight for fast entry
- **Card Forms**: Used in compact or mobile scenarios

Task 3: Run and Test the App

1. From the App Designer, click **Play** (or open the app from the Apps list)
2. Test the navigation:

- View the **Inventory** table and click into a record
- Browse the **User** table and open a record
- Confirm the default forms and views display correctly

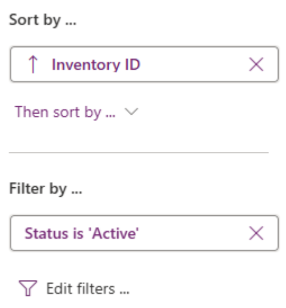


Inventory ID ↑	Name	Category	S
I-001	Dell Laptop XPS 13	Laptop	D
I-002	Epson Projector Z...	Projector	E
I-003	Logitech Webcam...	Accessory	L'
I-004	HP LaserJet Pro	Printer	H
INV-001	HP LaserJet Pro	Laptop	H

--- End of Task ---

Task 4: Customize Fields in Views (Optional)

1. Go to **Tables** > **Inventory** > **Views**
2. Select the view titled **Active Inventory** (or similar)
3. Click **Edit Columns**
   - Add or remove fields such as:
     - Serial Number
     - Inventory Status
     - Category
     - Purchase Date
4. Sort by field and Filter by field on the right hand panel



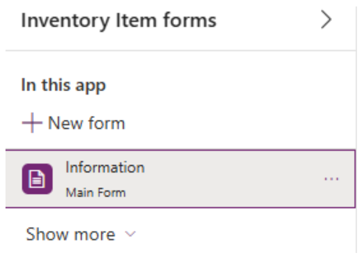
5. Save and Publish the view

Repeat this step as necessary for the **Employee** table (e.g., Full Name, Department, Email)

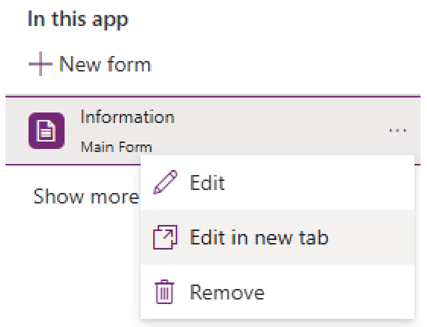
--- End of Task ---

Task 5: Customize the Form Layout (Optional)

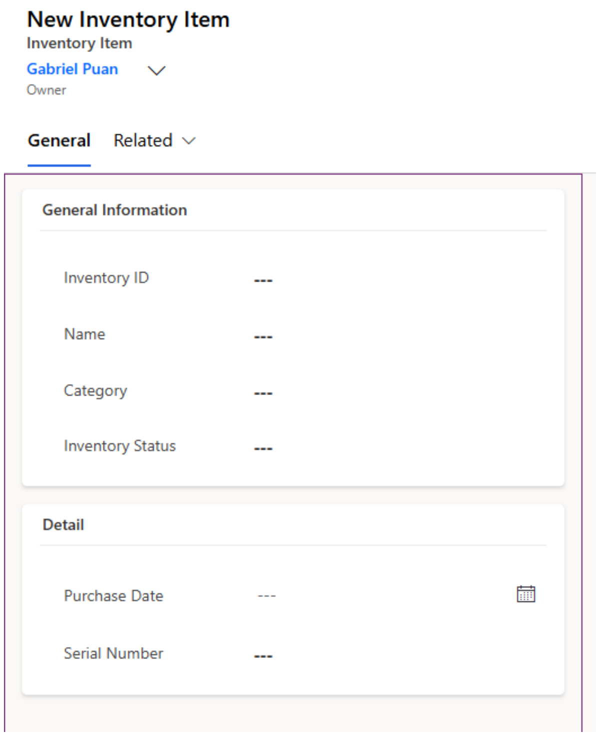
1. In the Model-driven app
2. Select **Inventory Item** form. See forms available on the right panel



3. Select **Edit in new tab** for the **Information Main Form** view



4. Rearrange the form layout to group related fields:
   - **General Information** Section: Inventory ID, Name, Category, Inventory Status
   - **Details** Section: Purchase Date, Serial Number


New Inventory Item
Inventory Item
Gabriel Puan
Owner

General | Related

General Information

Inventory ID	---
Name	---
Category	---
Inventory Status	---

Detail

Purchase Date	---
Serial Number	---

5. You can also:
   - Change field labels
   - Add or remove unused fields
   - Insert a horizontal tab or section if needed
6. Save and **Publish** the form

Repeat for the **Employee** table to group fields like Full Name, Email, Department

Tips

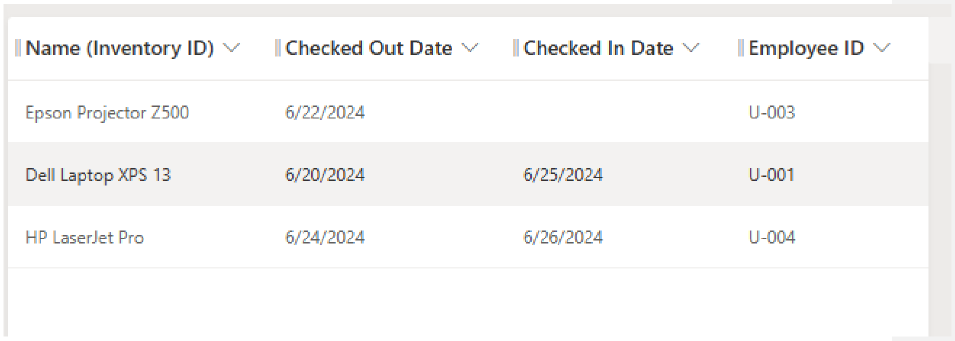
- Quick Create forms can be enabled via the table settings if you want faster record creation
- Add more tables (Service, CheckInOut) to expand the backend capabilities later

- Use security roles to control who can access this admin console

-- End of Task --

Task 6: Customize the Form View (Optional)

1. Navigate to the Solution
2. Table > CheckInOut Record > Create New View
3. Name View as Employee Active Record
4. Add these fields
   Inventory ID (Related), Checked Out Date, Checked In Date, Employee ID



Name (Inventory ID)	Checked Out Date	Checked In Date	Employee ID
Epson Projector Z500	6/22/2024		U-003
Dell Laptop XPS 13	6/20/2024	6/25/2024	U-001
HP LaserJet Pro	6/24/2024	6/26/2024	U-004

5. Save and Publish view