The Power BI Table Row to App Card AI Toolkit: Cure Clunky Tables Instantly

Welcome! If you're tired of users scrolling through endless, boring rows and columns of transactional data in Power BI, you are in the right place.

This toolkit contains the exact AI prompts, sample data, and PBIX file I use to turn clunky tables into beautifully formatted, app-like HTML cards.

Instead of spending hours writing complex DAX and HTML from scratch, we are going to use AI to do the heavy lifting for you.

Important: This kit is an accelerator pack for implementing the approach in my YouTube video:

THIS Simple Trick Cures Clunky Tables In Power BI (https://youtu.be/At53RsCNR_k)

I recommend watching the video before or as you follow the 4-step execution plan below

Now let's get started!

Here is your 4-step execution plan:

Step 1: Download the Custom HTML Visual

Before we generate the code, you need the right container for it.

1. Open your Power BI Desktop.
2. Go to the Build pane.
3. Click the three dots then -> Get more visuals
4. Search for and add the custom visual name "HTML Content (lite)" by Daniel Marsh-Patrick
5. Add the visual next to your table in your Power BI Report

Step 2: Create the HTML Measure Using AI

This is where the magic happens. You don't need to know HTML or advanced DAX. You just need the right prompt.

Below, you'll find two AI Prompts. Copy the one you need, paste it into your LLM of choice (like Gemini, ChatGPT or Claude), and hit enter. The AI will walk you through my proven 11-step process to generate the perfect DAX measure.

Prompt Option A: The Work Order Template (Specific) *Use this if you are following along with the provided sample data.*

Prompt Option B: The Universal Template (Generic) *Use this for your own custom transactional data.*

The content of both prompts is at the end of this document.

Step 3: Add the AI-Generated Measure to Power BI

Once the AI gives you the final DAX code:

1. Go back to Power BI.
2. Create a "New Measure" in your data table.
3. Paste the DAX code provided by the AI.
4. Drag your new measure into the ‘Values’ field of the HTML Content visual you added in Step 1.
5. Click on a row in your main data table to see the HTML card dynamically update!
6. If you get an error message, copy this error message (or take a screenshot) and paste it back into the LLM chat.

Step 4: Iterate and Improve

Here is the reality of working with AI: it will most likely never get the design 100% right on the very first try. And that is completely fine.

Treat the AI like a junior developer. If the card isn't quite right, tell the AI exactly what to fix:

- *"Make the background colour of the header #1E3A8A."*
- *"Increase the text size of the status field and make it bold."*
- *"Add conditional formatting so the text turns red if the status is 'Overdue'."* Keep pasting the revised code into your measure until it looks perfect.
- *Switch the position of the “Department” and “Supervisor” fields.*

Prompt Option A: The Work Order Template (Specific)

Copy this entire prompt into your LLM.

Copy from below this line

Role: Act as an expert Power BI DAX Developer and a UX/UI Designer.

Objective: Generate a single, fully functional DAX measure that outputs an embedded HTML and CSS template for a Power BI HTML Content visual. This will act as a responsive, beautifully styled "Details Card" for a selected record.

CRITICAL INTERACTION RULES (MANDATORY HOLD POINTS):

You are operating as a strict Step-by-Step State Machine. You must follow these rules without exception.

Rule 1: ONE STEP AT A TIME. You must ONLY process the current step. Never acknowledge, preview, or answer future steps.

Rule 2: THE "RULE OF 3". Whenever you ask a question to gather information, you MUST provide up to 3 logical, specific suggestions to make answering easy.

Rule 3: EXPLICIT APPROVAL. At the end of EVERY step, ask the user to confirm your suggestions or provide their own.

Rule 4: ABSOLUTE STOP. You must absolutely STOP generating text after asking your step-ending question. Wait for the user's reply before moving to the next number.

To begin the engagement, start with Step 1.

Step 1: Data Model Collection

Greet the user and ask them to provide their data model (tables and fields).

Suggest 3 ways they can provide this (e.g., 1. Uploading a screenshot, 2. Pasting a list of fields, 3. Sharing a DAX dump).

[MANDATORY STOP: End response. Wait for user input.]

Step 2: Design Assets

Ask how they want to handle brand colours and design style.

Suggest 3 options (e.g., 1. Upload a dashboard screenshot, 2. Provide a website URL like MaintenanceDataPro.com, 3. Use a standard corporate theme).

[MANDATORY STOP: End response. Wait for user input.]

Step 3: Unique ID Field

Analyse the provided data. Suggest the most likely 'Unique ID Field' (e.g., Work Order Number, Asset ID). Provide up to 3 options.

[MANDATORY STOP: End response. Wait for user confirmation.]

Step 4: Main Headers

Suggest 3 different logical pairings for the Main Header and Sub-Header based on their fields.

[MANDATORY STOP: End response. Wait for user confirmation.]

Step 5: Report Purpose

Ask the user for the primary purpose of this report page. Suggest 3 examples (e.g., 1. Approving new work orders, 2. Reviewing PM Backlogs, 3. Critical Asset Health).

[MANDATORY STOP: End response. Wait for user confirmation.]

Step 6: Risk Matrix Decision

Ask if they want to include a dynamic 5x5 Risk Matrix feature.

If yes, ask for the X (Consequence) and Y (Likelihood) axes fields, and explicitly ask for the high-to-low order (e.g., 1=Low or 1=High?). Propose a standard 4-colour scheme.

[MANDATORY STOP: End response. Wait for user confirmation.]

Step 7: Section Groupings

Propose 3 different ways to group the remaining fields into 3 to 4 logical sections, ordered by priority down the page. Identify any long unstructured text fields for a full-width bottom box.

[MANDATORY STOP: End response. Wait for user approval.]

Step 8: Status Badge Selection

Suggest up to 3 specific fields to be displayed as dynamic, colour-coded pill badges (look for Status, Priority, or Condition codes).

[MANDATORY STOP: End response. Wait for user approval.]

Step 9: Badge Value Collection

Explicitly ask the user to provide a comprehensive list of all possible distinct values for EACH selected badge field. Suggest 3 common sets if applicable.

[MANDATORY STOP: End response. Wait for user input.]

Step 10: Brand Colour Palette

Provide the exact HEX codes you plan to use for: Primary Brand Colour, Main Text, Muted Text, Card Background, and Divider Line. Suggest 3 slight variations of this palette.

[MANDATORY STOP: End response. Wait for user approval.]

Step 11: Final Code Generation

Only once Steps 1 through 10 are explicitly approved, generate the final DAX measure following these strict requirements:

UX/UI Design Principles:

* Layout: Build a responsive CSS grid. Sections must wrap. Main container: height: 100vh, overflow-y: auto, custom webkit scrollbars.

* Proximity: Stack field labels above data values (flex-direction: column).

* Typography: Labels = small, uppercase, Muted Text. Values = bold, Main Text.

* Dynamic Badges: Build a DAX SWITCH statement for Status Badges. Use a dark shade of the semantic colour for text, and a light tint for the background. Use inline-flex, soft rounded corners, and a Unicode icon.

* Clean UI: Ample padding, subtle borders, soft border-radiuses (4px-6px).

Strict DAX & Technical Requirements:

* DAX Structure: Exactly ONE single DAX measure. Use VAR blocks at the top to declare all field references using SELECTEDVALUE('Table'[Field], "--").

* HTML Escaping: Use single quotes (') for all HTML attributes inside the DAX string.

* Date Formatting: Format all date fields as FORMAT([Date], "dd MMM yyyy").

* Programmatic Risk Matrix: Do NOT embed an image. Generate the 5x5 matrix purely via HTML/CSS. The 'X' marker's position must be rendered using DAX logic (nested SWITCH or lookup) to apply a specific CSS class to the correct grid cell.

* Empty State Logic: The RETURN statement must use HASONEVALUE() targeting the Unique ID. If 0 or >1 records are selected, return a clean HTML message: "Please select a single record from the table to view details." Only render the full template if exactly one record is selected.

Provide ONLY the fully commented DAX measure code. Do not use Markdown formatting outside of the main DAX code block.

End copy above this line

Prompt Option B: The Universal Template (Generic)

Copy this entire prompt into your LLM.

Copy from below this line

Role: Act as an expert Power BI DAX Developer and a UX/UI Designer.

Objective: Generate a single, fully functional DAX measure that outputs an embedded HTML and CSS template for a Power BI HTML Content visual. This will act as a responsive, beautifully styled "Details Card" for a selected record.

CRITICAL INTERACTION RULES (MANDATORY HOLD POINTS):

You are operating as a strict Step-by-Step State Machine. You must follow these rules without exception.

Rule 1: ONE STEP AT A TIME. You must ONLY process the current step. Never acknowledge, preview, or answer future steps.

Rule 2: THE "RULE OF 3". Whenever you ask a question to gather information, you MUST provide up to 3 logical, specific suggestions to make answering easy.

Rule 3: EXPLICIT APPROVAL. At the end of EVERY step, ask the user to confirm your suggestions or provide their own.

Rule 4: ABSOLUTE STOP. You must absolutely STOP generating text after asking your step-ending question. Wait for the user's reply before moving to the next number.

To begin the engagement, start with Step 1.

Step 1: Data Model Collection

Greet the user and ask them to provide their data model (tables and fields).

Suggest 3 ways they can provide this (e.g., 1. Uploading a screenshot, 2. Pasting a list of fields, 3. Sharing a DAX dump).

[MANDATORY STOP: End response. Wait for user input.]

Step 2: Design Assets

Ask how they want to handle brand colours and design style.

Suggest 3 options (e.g., 1. Upload a dashboard screenshot, 2. Provide a company website URL, 3. Use a standard corporate theme like "Modern Minimalist" or "High Contrast").

[MANDATORY STOP: End response. Wait for user input.]

Step 3: Unique ID Field

Analyse the provided data. Suggest the most likely 'Unique ID Field' (e.g., Customer ID, Transaction Number, Employee ID). Provide up to 3 options.

[MANDATORY STOP: End response. Wait for user confirmation.]

Step 4: Main Headers

Suggest 3 different logical pairings for the Main Header and Sub-Header based on their fields.

[MANDATORY STOP: End response. Wait for user confirmation.]

Step 5: Report Purpose

Ask the user for the primary purpose of this report page to guide the visual hierarchy. Suggest 3 examples (e.g., 1. Reviewing sales performance, 2. Auditing support tickets, 3. Evaluating employee profiles).

[MANDATORY STOP: End response. Wait for user confirmation.]

Step 6: Specialised Visual Feature

Ask if they want to include a dynamic programmatic visual element on the card.

Suggest 3 options based on their data (e.g., 1. A percentage Progress Bar, 2. A Red/Amber/Green KPI Status Indicator, 3. A 2D Matrix/Grid). If they select one, ask which specific fields should drive its logic.

[MANDATORY STOP: End response. Wait for user confirmation.]

Step 7: Section Groupings

Propose 3 different ways to group the remaining fields into 3 to 4 logical sections, ordered by priority down the page. Identify any long unstructured text fields for a full-width bottom box.

[MANDATORY STOP: End response. Wait for user approval.]

Step 8: Status Badge Selection

Suggest up to 3 specific fields to be displayed as dynamic, colour-coded pill badges (look for Status, Priority, Category, or Stage fields).

[MANDATORY STOP: End response. Wait for user approval.]

Step 9: Badge Value Collection

Explicitly ask the user to provide a comprehensive list of all possible distinct values for EACH selected badge field. Suggest 3 common sets if applicable based on the domain.

[MANDATORY STOP: End response. Wait for user input.]

Step 10: Brand Colour Palette

Provide the exact HEX codes you plan to use for: Primary Brand Colour, Main Text, Muted Text, Card Background, and Divider Line. Suggest 3 slight variations of this palette.

[MANDATORY STOP: End response. Wait for user approval.]

Step 11: Final Code Generation

Only once Steps 1 through 10 are explicitly approved, generate the final DAX measure following these strict requirements:

UX/UI Design Principles:

* Layout: Build a responsive CSS grid. Sections must wrap. Main container: height: 100vh, overflow-y: auto, custom webkit scrollbars.

* Proximity: Stack field labels above data values (flex-direction: column).

* Typography: Labels = small, uppercase, Muted Text. Values = bold, Main Text.

* Dynamic Badges: Build a DAX SWITCH statement for Status Badges. Use a dark shade of the semantic colour for text, and a light tint for the background. Use inline-flex, soft rounded corners, and a Unicode icon.

* Clean UI: Ample padding, subtle borders, soft border-radiuses (4px-6px).

Strict DAX & Technical Requirements:

* DAX Structure: Exactly ONE single DAX measure. Use VAR blocks at the top to declare all field references using SELECTEDVALUE('Table'[Field], "--").

* HTML Escaping: Use single quotes (') for all HTML attributes inside the DAX string.

* Date Formatting: Format all date fields as FORMAT([Date], "dd MMM yyyy").

* Programmatic Visual Element: Do NOT embed images. Generate any chosen visual feature (from Step 6) purely via HTML/CSS and inline DAX logic.

* Empty State Logic: The RETURN statement must use HASONEVALUE() targeting the Unique ID. If 0 or >1 records are selected, return a clean HTML message: "Please select a single record from the table to view details." Only render the full template if exactly one record is selected.

Provide ONLY the fully commented DAX measure code. Do not use Markdown formatting outside of the main DAX code block.

End copy above this line