**n8n Workflow Guide: AI-Powered Dashboard Design**

This guide will walk you through setting up an n8n workflow that reads data from a CSV file, summarizes it, and uses an AI agent to generate a dashboard specification based on the data.

**Workflow Overview:**

The workflow consists of the following steps:

1. **Trigger:** Start the workflow manually.
2. **Set Analysis Info:** Define input parameters like CSV file path, description, and dashboard identifiers.
3. **Read/Write Files from Disk:** Read the CSV file from your local disk.
4. **Extract from File:** Parse the CSV data into a structured format.
5. **Create Summary:** Generate a summary of the CSV data using Python code.
6. **Summary to string:** Convert the summary data into a string format.
7. **Merge:** Combine the summary and input parameters.
8. **OpenAI Chat Model:** Use an OpenAI model to process the data and generate a dashboard specification.
9. **AI Agent:** Use an AI Agent to structure the prompt and interact with the OpenAI model.
10. **Create Dashboard:** (Placeholder) Represents the action to create a dashboard using the generated specification.

**Step-by-Step Instructions:**

1. **Add a Trigger Node:**
   * Drag and drop a "When clicking 'Test workflow'" trigger node onto the canvas. This will allow you to manually start the workflow for testing.
   * No configuration is needed for this node.
2. **Add a "Set Analysis Info" Node:**
   * Drag and drop a "Set" node after the trigger node. Rename it to "Set Analysis Info".
   * In the "Parameters" tab, select "Mode" as "Manual Mapping".
   * Add the following fields by clicking "Add Field" and configure them as shown in the screenshot:
     + **Field Name:** csv\_file
       - **Value:** Enter the full path to your CSV file on your local machine. For example: E:/dev/dgn8r/football/sales-data-sample.csv (Make sure this path is accessible to your n8n instance).
     + **Field Name:** description
       - **Value:** Provide a brief description of your CSV data and the goal for the dashboard. For example: This file contains sales data for different regional offices. I want a focus on area and geographic maps.
     + **Field Name:** dashboard\_id
       - **Value:** Give a name to your dashboard. For example: Sales dashboard
     + **Field Name:** user\_id
       - **Value:** Enter a user identifier. For example: Alex

**Common Check:**

* + **File Path:** Ensure the csv\_file path is correct and the file exists at that location. n8n needs to be able to access this file on the server where it's running.
  + **Description Clarity:** The description field is important for the AI. Make it clear and concise to guide the dashboard design.

1. **Add a "Read/Write Files from Disk" Node:**
   * Drag and drop a "Read/Write Files from Disk" node after the "Set Analysis Info" node. Rename it to "Read/Write Files from Disk".
   * In the "Parameters" tab, configure the following:
     + **Operation:** Select "Read File(s) From Disk".
     + **File(s) Selector:** Use an expression to dynamically get the file path from the previous node. Click the variable selector ( fx ) next to the field and choose:
       - Node -> Set Analysis Info -> JSON -> csv\_file
       - This will populate the field with the expression: {{ $json.csv\_file }}

**Common Check:**

* + **Permissions:** Ensure n8n has the necessary permissions to read files from the specified directory.

1. **Add an "Extract from File" Node:**
   * Drag and drop an "Extract from File" node after the "Read/Write Files from Disk" node. Rename it to "Extract from File".
   * In the "Parameters" tab, configure the following:
     + **Operation:** Select "Extract From CSV".
     + **Input Binary Field:** Select "data" from the dropdown. This refers to the binary data output from the "Read/Write Files from Disk" node.

**Optional Configurations (If needed):**

* + **Delimiter:** If your CSV uses a delimiter other than a comma (,), specify it here.
  + **Header Row:** Ensure "Header Row" is checked if your CSV file has a header row.

1. **Add a "Create Summary" Node:**
   * Drag and drop a "Code" node after the "Extract from File" node. Rename it to "Create Summary".
   * In the "Parameters" tab, set:
     + **Language:** Select "Python (Beta)".
   * In the "Code" section, paste the following Python code:
   * import json
   * data = items[0].json
   * summary = data[:5] # Taking the first 5 rows as a summary. Adjust as needed.
   * summary\_string = json.dumps(summary) # Convert summary to JSON string for easier use later

return [{"summary": summary\_string}]

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487).Python

1. **Explanation of the Python Code:**
   * data = items[0].json: Accesses the JSON data output from the "Extract from File" node (which contains your CSV data).
   * summary = data[:5]: Takes the first 5 rows of the data as a summary. You can modify this to create a more comprehensive summary if needed (e.g., calculate averages, unique values, etc.).
   * summary\_string = json.dumps(summary): Converts the Python list of dictionaries (summary) into a JSON string. This makes it easier to pass into the AI prompt later.
   * return [{"summary": summary\_string}]: Returns the summary as a new item with a field called "summary".

**Common Check:**

* + **Python Environment:** Ensure your n8n instance has Python enabled and the necessary libraries are installed if you use more complex Python code.
  + **Summary Logic:** Adjust the Python code to create a summary that is informative for the AI agent. For very large CSVs, summarizing key statistics might be more efficient than sending raw data.

1. **Add a "Summary to string" Node:**
   * Drag and drop a "Set" node after the "Create Summary" node. Rename it to "Summary to string".
   * In the "Parameters" tab, select "Mode" as "Manual Mapping".
   * Add a field:
     + **Field Name:** summary
     + **Value:** Use an expression to get the summary from the previous node. Click the variable selector ( fx ) and choose:
       - Node -> Create Summary -> JSON -> summary
       - This will populate the field with the expression: {{ $json.summary }}
       - Set the "Value Type" to "String".

**Why this node?** Although the summary is already a string (JSON string), this node ensures it's explicitly treated as a string in n8n and makes it easier to reference in the next steps.

1. **Add a "Merge" Node:**
   * Drag and drop a "Merge" node after the "Summary to string" node. Rename it to "Merge".
   * In the "Parameters" tab, set "Mode" to "SQL Query".
   * Set "Number of Inputs" to 2.
   * In the "Query" section, paste the following SQL query:
   * SELECT
   * \*
   * FROM

INPUT1 LEFT JOIN INPUT2 ON input1.name = input2.id

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487).SQL

1. **Explanation of the Merge Node:**
   * This "Merge" node is used to combine the output from the "Summary to string" node (INPUT1) and the "Set Analysis Info" node (INPUT2) into a single item.
   * The SQL query SELECT \* FROM INPUT1 LEFT JOIN INPUT2 ON input1.name = input2.id effectively performs a left join, but in this context, it's primarily used to combine the fields from both input items into a single output item for the next node. The ON condition is not strictly necessary for this simple combination but is required by the "SQL Query" mode.
2. **Add an "OpenAI Chat Model" Node:**
   * Drag and drop an "OpenAI Chat Model" node after the "Merge" node. Rename it to "OpenAI Chat Model".
   * In the "Parameters" tab, configure the following:
     + **Credential to connect with:** Select or create your OpenAI API credential. You'll need an OpenAI API key.
     + **Model:** Choose a suitable model, like gpt-4o or gpt-3.5-turbo. gpt-4o is generally more capable but might be more expensive.
3. **Add an "AI Agent" Node:**
   * Drag and drop an "AI Agent" node after the "OpenAI Chat Model" node. Rename it to "AI Agent".
   * In the "Parameters" tab, configure the following:
     + **Agent:** Select "Tools Agent".
     + **Tools Agent:** Leave this as is (no tools are needed in this example).
     + **Source for Prompt (User Message):** Select "Define below".
     + **Text:** Paste the following prompt into the "Text" area. **Crucially, replace the placeholders {{ $json.summary }} and {{ $json.description }} with the expressions as shown below**:
   * You are an AI assistant helping design dashboards for a data visualization tool (vizro-ai).
   * Based on this information, propose an interesting dashboard layout with multiple pages, each describing a specific visualization. Use this format for each page, inspired by the example below (this is just an example and doesn't relate to our data)
   * Example to return
   * "I need a page showing 1 card and 1 chart.
   * The card says 'The Gapminder dataset provides historical data on countries' development indicators.'
   * The chart is a scatter plot showing GDP per capita vs. life expectancy.
   * GDP per capita on the x-axis, life expectancy on the y-axis, and colored by continent.
   * Layout the card on the left and the chart on the right. The card takes 1/3 of the whole space on the left.
   * The chart takes 2/3 of the whole space and is on the right.
   * Add a filter to filter the scatter plot by continent.
   * Add a second filter to filter the chart by year."
   * Ensure the dashboard uses the data fields from the CSV (e.g., 'name', 'age', 'city') creatively to create insightful visualizations. Propose 1 page total, each with unique and engaging layouts. Return only the dashboard specification in the format above, with no explanation or additional text. You can use the cards to write static data / headers etc
   * Only return a single result item to specify a single dashboard page
   * What follows is the CSV with example summary data:
   * {{ $json["Summary to string"].summary }}
   * The user has provided this high-level description of the data:

{{ $json["Set Analysis Info"].description }}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487).

* + - **Require Specific Output Format:** Enable this toggle.

**Important Prompt Details:**

* + **Context:** The prompt clearly defines the AI's role and the desired output format.
  + **Example Format:** The example provides a clear structure for the AI to follow when generating the dashboard specification.
  + **Data Fields Instruction:** The prompt explicitly instructs the AI to use the CSV data fields creatively.
  + **Summary and Description Injection:**
    - {{ $json["Summary to string"].summary }}: This expression dynamically inserts the JSON string summary created in the "Summary to string" node into the prompt. **Make sure the node name "Summary to string" matches your node's name exactly.**
    - {{ $json["Set Analysis Info"].description }}: This expression dynamically inserts the description you provided in the "Set Analysis Info" node. **Make sure the node name "Set Analysis Info" matches your node's name exactly.**

**Common Checks:**

* + **OpenAI API Key:** Ensure your OpenAI API key is valid and you have sufficient credits.
  + **Prompt Accuracy:** Double-check the prompt text, especially the expressions for injecting the summary and description. Typos in node names will cause errors.
  + **Model Choice:** Experiment with different OpenAI models if the initial results are not satisfactory. gpt-4o is generally better for complex tasks but might be overkill for simple dashboard layouts.

1. **Connect "AI Agent" Output to "OpenAI Chat Model" Input:**
   * Drag a connection from the output of the "AI Agent" node (the little circle on the right) to the input of the "OpenAI Chat Model" node (the little circle on the left).
   * Select the "messages" input from the connection options that appear. This will send the processed prompt from the AI Agent to the OpenAI model.
2. **Add a "Create Dashboard" Node (Placeholder):**
   * Drag and drop an "HTTP Request" node after the "OpenAI Chat Model" node. Rename it to "Create Dashboard".
   * This node is a placeholder. In a real application, you would configure this node to send a POST request to your dashboard creation API using the dashboard specification generated by the AI.
   * For demonstration purposes, you can configure it as shown in the screenshot:
     + **Method:** POST
     + **URL:** http://127.0.0.1:8000/create\_dashboard (Replace with your actual API endpoint if you have one)
     + **Send Body:** Enable
     + **Body Content Type:** JSON
     + **Body:**
       - Use an expression to pass the text output from the "OpenAI Chat Model" node as the dashboard\_spec in the JSON body:
       - {
       - "dashboard\_spec": "{{ $json.response.generations[0].text }}",
       - "dashboard\_id": "{{ $json["Set Analysis Info"].dashboard\_id }}",
       - "user\_id": "{{ $json["Set Analysis Info"].user\_id }}"

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487).Json

1. ![alt text](https://i.imgur.com/o3Y87Vd.png)
2. **Explanation of the "Create Dashboard" Placeholder Node:**
   * This node simulates sending the dashboard specification to a hypothetical API endpoint to create the dashboard.
   * **dashboard\_spec**: Extracts the generated dashboard specification text from the OpenAI model's response.
   * **dashboard\_id and user\_id**: Passes through the dashboard ID and user ID from the initial "Set Analysis Info" node.

**Running the Workflow and Testing:**

1. **Save your workflow.**
2. Click the "Test workflow" button in the trigger node ("When clicking 'Test workflow'").
3. Observe the output of each node as the workflow executes.
4. Check the output of the "OpenAI Chat Model" node. It should contain the dashboard specification in the desired format.
5. Examine the output of the "Create Dashboard" node (if you have a real API endpoint configured).

**Common Gotchas and Troubleshooting:**

* **API Keys and Credentials:** Double-check your OpenAI API key and n8n credentials.
* **File Paths:** Verify the CSV file path in the "Set Analysis Info" and "Read/Write Files from Disk" nodes.
* **Node Naming in Expressions:** Ensure the node names in your expressions (e.g., {{ $json["Summary to string"].summary }}) exactly match the names of your nodes. Typos are a common source of errors.
* **Prompt Quality:** If the generated dashboard specifications are not satisfactory, refine the prompt in the "AI Agent" node. Be more specific about your requirements, data fields, and desired visualizations.
* **Error Messages:** Carefully read any error messages in n8n. They usually provide clues about what went wrong. Check the "Execution Log" for more detailed error information.
* **Rate Limits:** Be mindful of OpenAI API rate limits, especially when testing frequently.

**Customization and Next Steps:**

* **Improve Summary Logic:** Enhance the Python code in the "Create Summary" node to generate more insightful summaries of your data.
* **Refine the Prompt:** Experiment with different prompts to guide the AI agent towards generating more specific and creative dashboard designs.
* **Implement Real Dashboard Creation:** Replace the placeholder "Create Dashboard" node with a connection to your actual dashboard creation API or tool.
* **Add Error Handling:** Implement error handling in your workflow to gracefully manage potential issues like file not found, API errors, etc.

By following these steps, you should be able to recreate and customize this n8n workflow to generate dashboard specifications using AI. Remember to test thoroughly and iterate on your prompt and workflow to achieve the desired results!