SAP data analytics master class
Special Python Meeting- ChatGPT API

17th June 2026


Using API to talk with ChatGPT

WHAT WILL WE COVER TODAY?

01 *Create a client*

02 *Create an assistant*

03 *Load tasks from JSON*

04 *Create AI prompt*

05 *Process task*

06 *Create chatbot*

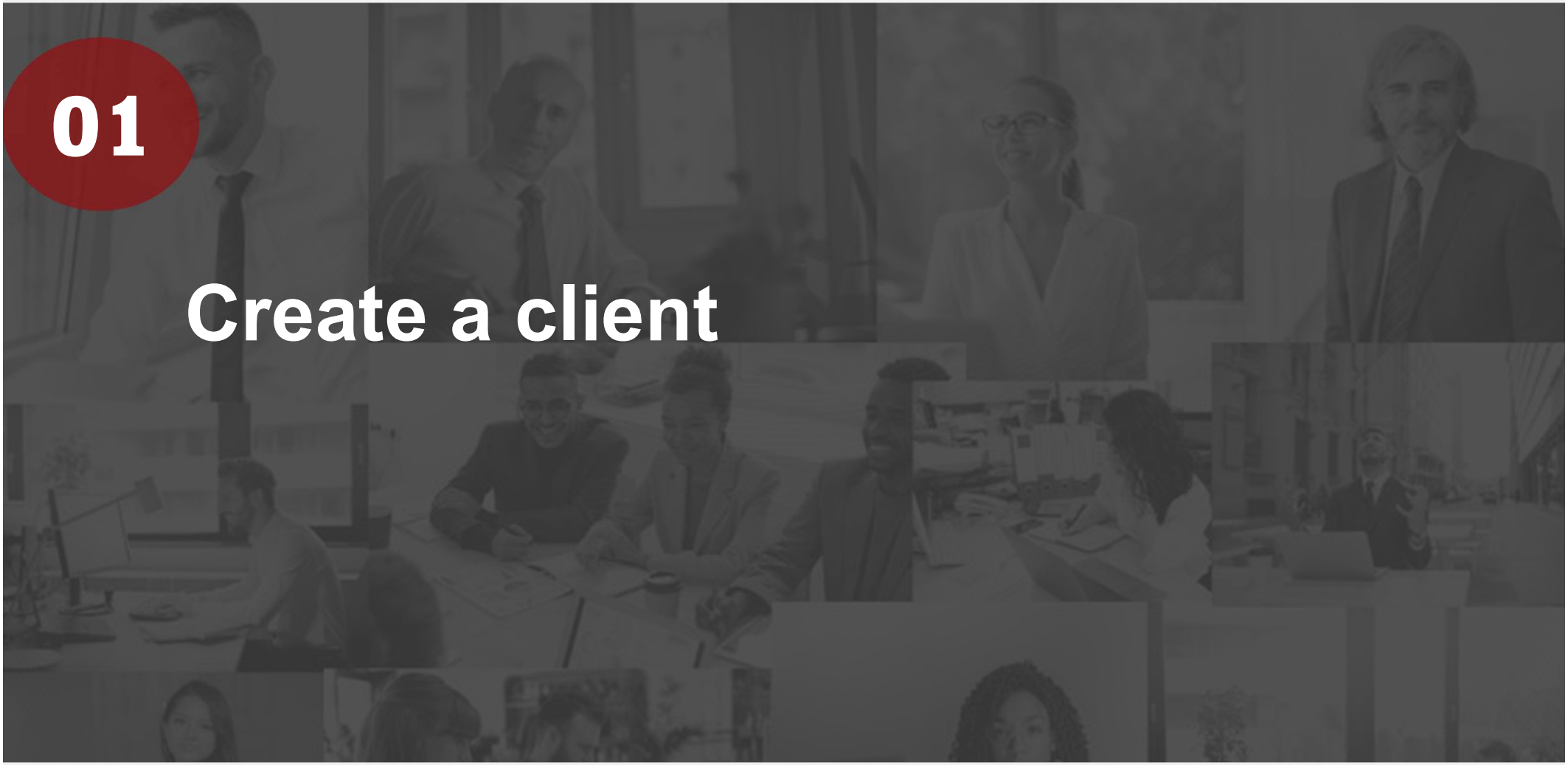
07 *Call the chatbot*

Already doing APIs?

20260617_01_AlreadyDoingAPIs?

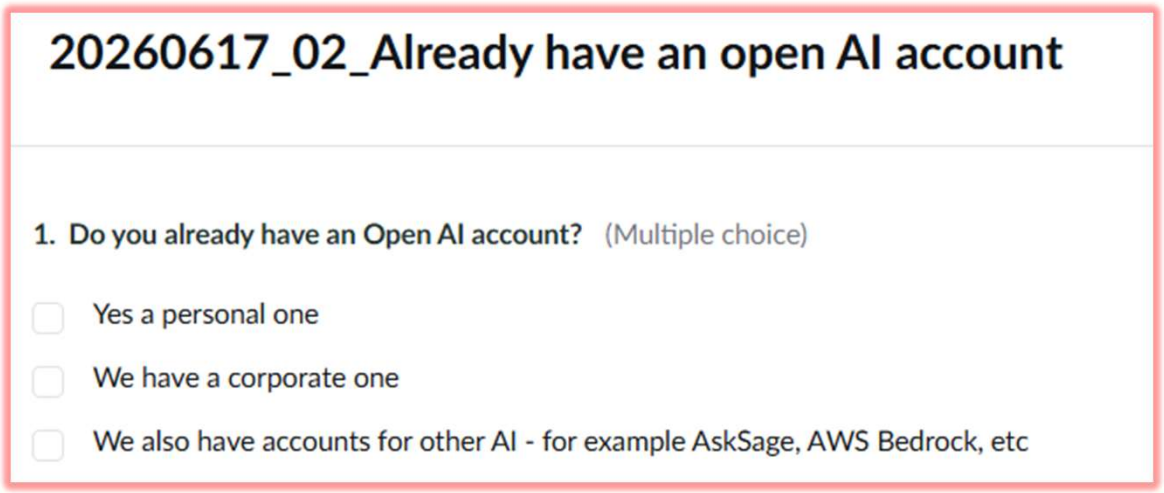
1. Are you already doing APIs with python? (Single choice)

- Yes
- No


01
Create a client

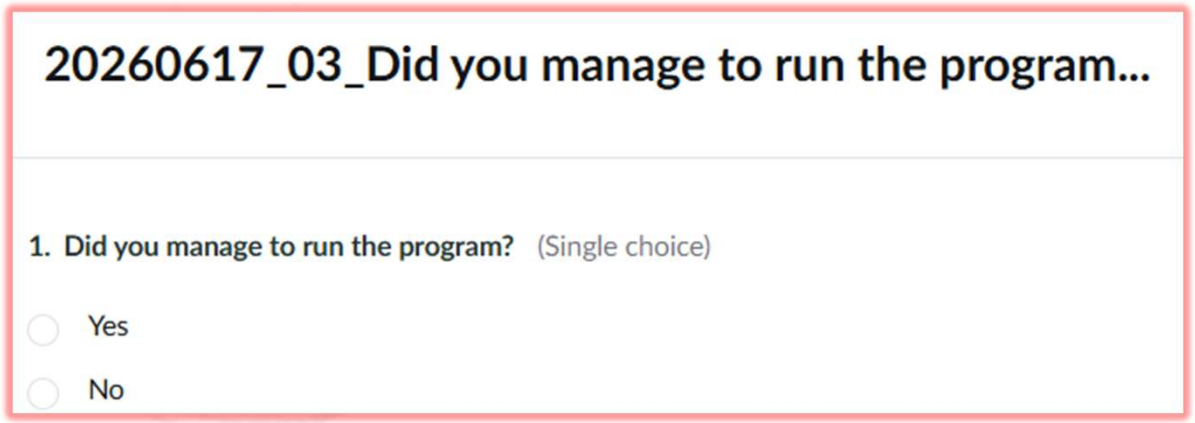
Already have an open AI account?

Do you already have an open AI account?



Manage to run the program already?

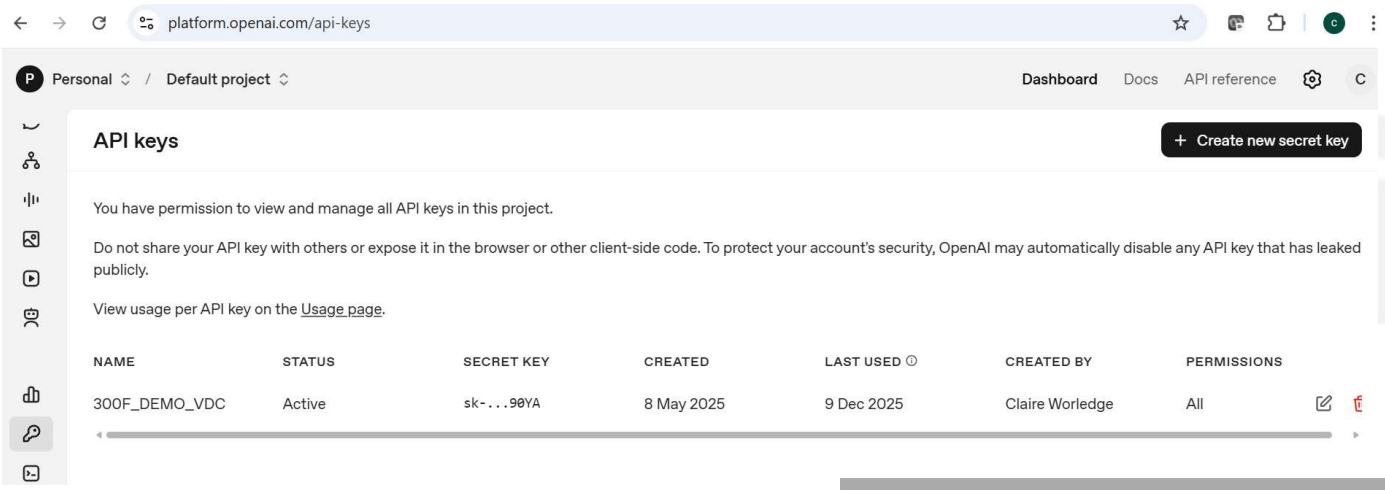
Did you manage to run the program before the meeting?

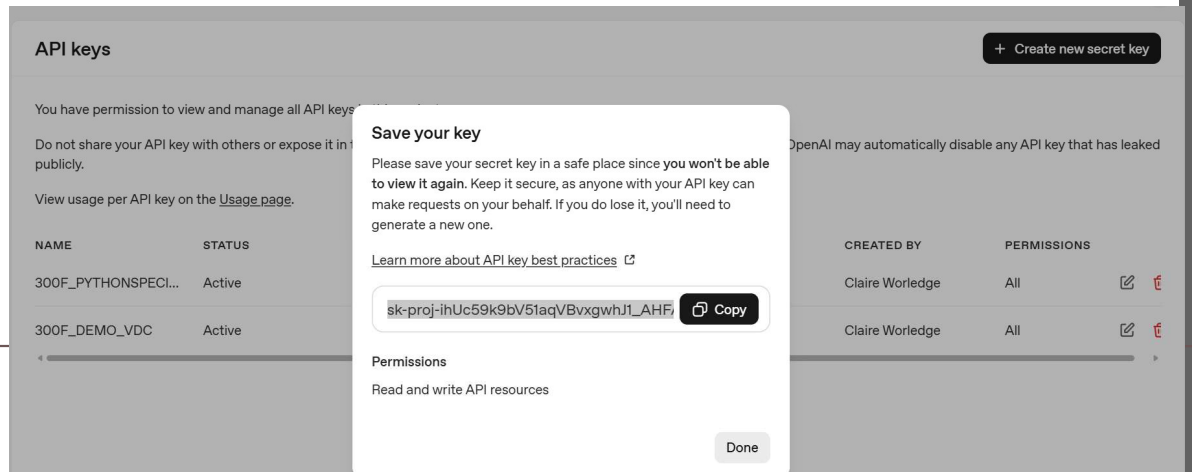


Create an API key

If you want to create an API to chat GPT, the first thing that you need to do is to create a key:

https://platform.openai.com/api-keys





Now we have automated all the variables

The only variable that you need to create is your openAI secret key in .env.
The folder paths are now picked up automatically.
Requirements are also run automatically.

02_PROGRAMMES > Z00_OPENAI_API.py > ...

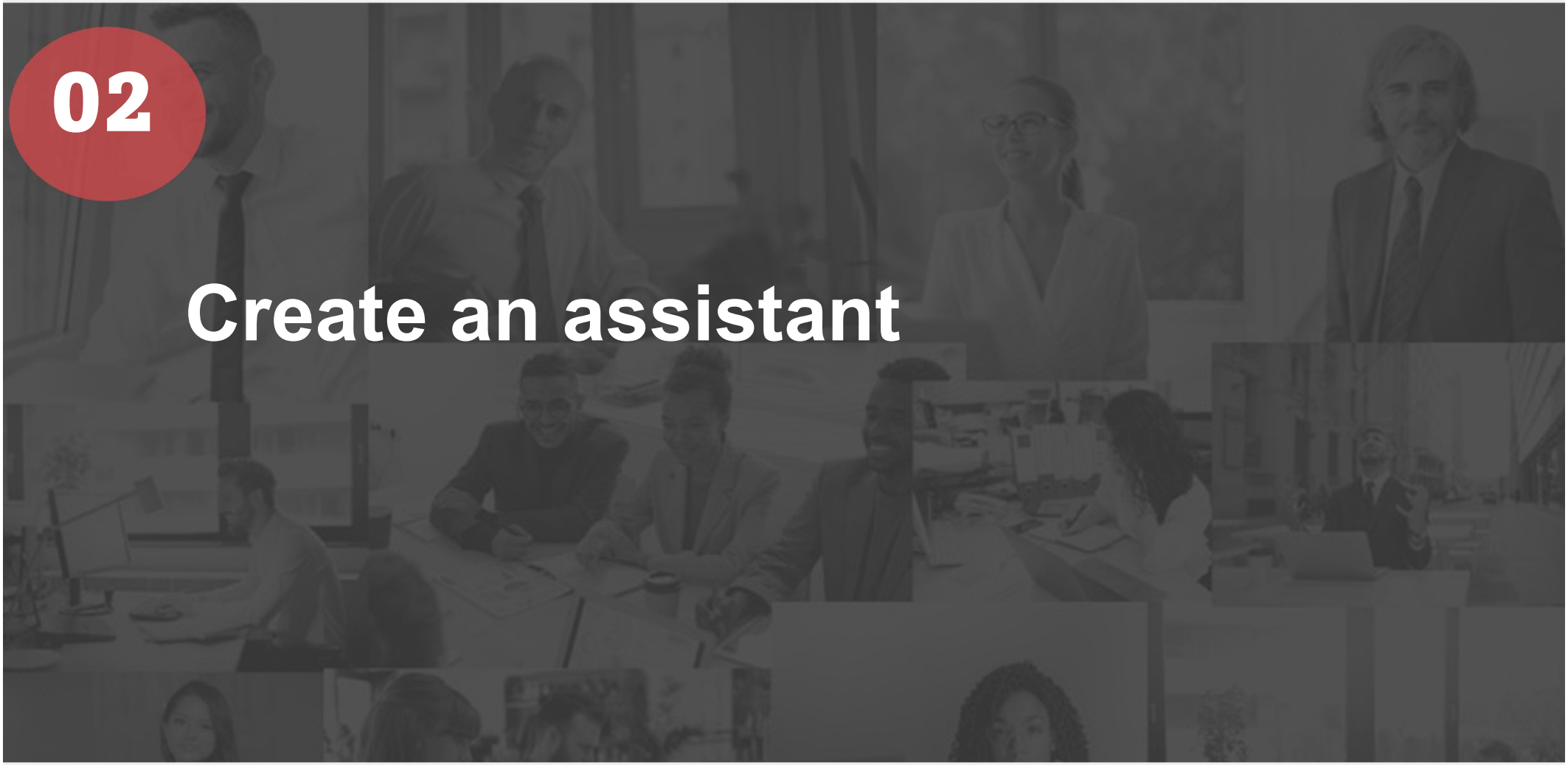
```python
# Script objective:
# Use openAI to give feedback on analytics results

# See README.md for set-up instructions

# 1/ Set-up
# 1.1/ Find programs folder, project root folder

import sys as PI_SYS
import os as PI_OS

ZV_ST_02PROG_FOLDER = PI_OS.path.dirname(
    PI_OS.path.abspath(__file__)
)

ZV_ST_ROOT_FOLDER = PI_OS.path.dirname(
    ZV_ST_02PROG_FOLDER
)

# 1.2/ Add programs folder to system path
#      (enables import from Z_SHARED_FUNCTIONS)
PI_SYS.path.append(ZV_ST_02PROG_FOLDER)
```

```python
# 1.4/ Import variables from AM_VARIABLES and .env to a dictionary in RAM
from Z_SHARED_FUNCTIONS.FC_CREATE_DI_VARIABLES import FC_CREATE_DI_VARIAB
ZV_DI_VARIABLES= FC_CREATE_DI_VARIABLES(ZVFCI_ST_ROOT_FOLDER=ZV_ST_ROOT_F

# 1.5/ Import libraries
from openai import OpenAI as PI_OPENAI
import os as PI_OS
import json as PI_JSON
import polars as PI_POLARS
from openai import AuthenticationError as PI_AUHENTICAIONERROR

# 1.6/ Import other custom functions
from Z_SHARED_FUNCTIONS.FC_EXPORT import FC_EXPORT_EXCEL_POLARS

# 2/ Variables
ZV_OPENAI_API_KEY = ZV_DI_VARIABLES.get('ZV_OPENAI_API_KEY')
ZV_ST_SOURCES_FOLDER = PI_OS.path.join(
        ZV_ST_ROOT_FOLDER,
        '01_SOURCES'
    )
ZV_ST_RESULTS_FOLDER = PI_OS.path.join(
        ZV_ST_ROOT_FOLDER,
        '03_RESULTS'
    )

ZV_ST_SOURCE_FILE_NAME=ZV_DI_VARIABLES.get('ZV_ST_SOURCE_FILE_NAME')
ZV_ST_JSON_FILE_NAME=ZV_DI_VARIABLES.get('ZV_ST_JSON_FILE_NAME')
ZV_ST_RESULTS_FILE_NAME = ZV_DI_VARIABLES.get('ZV_ST_RESULTS_FILE_NAME')
```

Create a client

Once we have the key, then we can use the openAI library to create a client:

```
58  # 3/ ChatGPT API function
59  def Z00_OPENAI_API():
60
61      def FC_CREATE_AI_CLIENT():
62          ZV_OB_OPENAI_CLIENT = PI_OPENAI(
63              default_headers={"OpenAI-Beta": "assistants=v2"},
64              api_key=ZV_ST_OPENAI_API_KEY
65          )
66                                                  (property) beta: Beta
67          ZV_OB_OPENAI_THREAD = ZV_OB_OPENAI_CLIENT.beta.threads.create()
68          ZV_ST_THREAD_ID = ZV_OB_OPENAI_THREAD.id
69
70          return ZV_OB_OPENAI_CLIENT, ZV_ST_THREAD_ID
71
```


02
Create an assistant

Create an assistant

Once we have created a client, we can use the client to create an assistant:

Tip:

Use markdown to get a better response: https://commonmark.org

https://spec.commonmark.org/

```
def FC_CREATE_AI_ASSISTANT(ZVFCI_AI_CLIENT):
    ZV_OB_ASSISTANT = ZVFCI_AI_CLIENT.beta.assistants.create(
        name="SAP Dashboard Risk Analyst",
        instructions='''
        You are a skilled Data Analyst analyzing SAP dashboards (Order to Cash and others). Y

        1. **Observations**:
        - Review charts for patterns, anomalies, or inefficiencies.

        2. **Risk Assessment**:
        - For each issue, output a Markdown table with:
            - **Issue** (≤10 words)
            - **Priority** (High/Medium/Low)
            - **Observation**
            - **Risk**
            - **Recommendation** (with audit test suggestions)

        - Example:
        ```
Issue	Priority	Observation	Risk	Recommendati
Excess Billing	High	Billing > Orders	Revenue inflation risk	Review perio
        ```

        3. **Executive Summary**:
        - Summarize all findings clearly for audit teams and senior stakeholders.

        CRITICAL: You MUST include all three sections (### Observations, ### Risk Assessment,
        Always respond in **Markdown table format**. If formatting fails, retry.
        ''',
        model="gpt-4o-mini",
        tools=[]
    )
    return ZV_OB_ASSISTANT

```


03
Load tasks from JSON

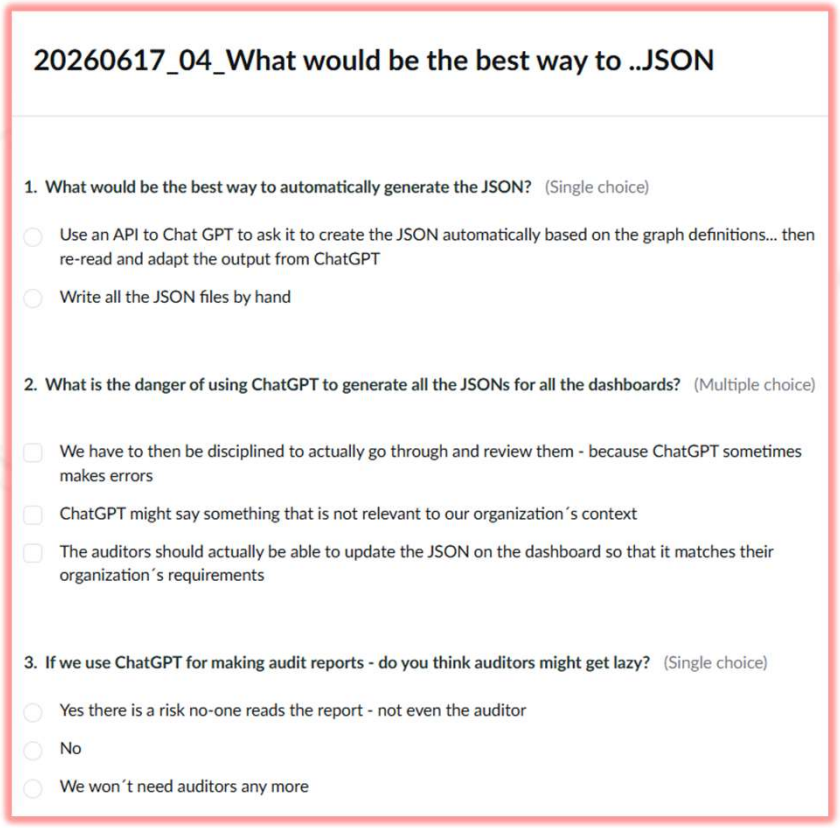
Load tasks from JSON

Let’s take a quick look at our JSON file!

```
108
109     def FC_LOAD_TASKS_FROM_JSON(ZVFCI_ST_JSON_FILE_PATH, ZVFCI_ST_JSON_NAME):
110         ZV_ST_FILE_PATH = PI_OS.path.join(ZVFCI_ST_JSON_FILE_PATH, ZVFCI_ST_JSON_NAME)
111
112         if not PI_OS.path.isfile(ZV_ST_FILE_PATH):
113             raise FileNotFoundError(f"Do not find: {ZV_ST_FILE_PATH}")
114
115         with open(ZV_ST_FILE_PATH, 'r') as file:
116             ZV_DI_ALL_TASKS = PI_JSON.load(file)
117
118         return ZV_DI_ALL_TASKS
119
```

What would be the best way to create the JSON for all 300 dashboards?

What do you think would be the best way to create a JSON file for all 300 dashboards?



20260617_04_What would be the best way to ..JSON

1. **What would be the best way to automatically generate the JSON?** (Single choice)

- Use an API to Chat GPT to ask it to create the JSON automatically based on the graph definitions... then re-read and adapt the output from ChatGPT
- Write all the JSON files by hand

2. **What is the danger of using ChatGPT to generate all the JSONs for all the dashboards?** (Multiple choice)

- We have to then be disciplined to actually go through and review them - because ChatGPT sometimes makes errors
- ChatGPT might say something that is not relevant to our organization´s context
- The auditors should actually be able to update the JSON on the dashboard so that it matches their organization´s requirements

3. **If we use ChatGPT for making audit reports - do you think auditors might get lazy?** (Single choice)

- Yes there is a risk no-one reads the report - not even the auditor
- No
- We won´t need auditors any more

04

Create AI prompt

Create AI prompt

In 300Framework, we grab the information from the chart, the database and the user message and the JSON instructions.. but here we do a standalone version with variables.

```python

def FC_CREATE_AI_PROMPT(
    ZVFCI_ST_DATA_FILE_PATH,
    ZVFCI_ST_DATA_FILE_NAME,
    ZVFCI_DI_TASK
):
    ZV_ST_FILE_PATH = PI_OS.path.join(ZVFCI_ST_DATA_FILE_PATH, ZVFCI_ST_DATA_FILE_N
    ZV_DF_DATA = PI_POLARS.read_csv(ZV_ST_FILE_PATH)
    ZV_DI_DATA = ZV_DF_DATA.to_dict(as_series=False)

    ZV_ST_TASK_NAME = ZVFCI_DI_TASK.get('task_name', 'Unknown Task')
    ZV_ST_TASK_TITLE = ZVFCI_DI_TASK.get('task_title', 'Unknown Chart')
    ZV_ST_TASK_CONTENT = ZVFCI_DI_TASK.get('task_content', '')

    ZV_ST_PROMPT = (
        f"Analysis Task: {ZV_ST_TASK_NAME}\n"
        f"Chart Title: {ZV_ST_TASK_TITLE}\n\n"
        f"Task Instructions:\n{ZV_ST_TASK_CONTENT}\n\n"
        f"Data:\n{PI_JSON.dumps(ZV_DI_DATA, indent=2)}\n\n"
        f"Please provide your response with:\n"
        f"1. ### Observations section\n"
        f"2. ### Risk Assessment section (as a markdown table)\n"
        f"3. ### Executive Summary section\n"
    )

    # print('ZV_ST_PROMPT', ZV_ST_PROMPT)

    return ZV_ST_PROMPT, ZV_ST_TASK_TITLE
```

05

Process task

Process task

Add the thread

Run the query

Get the response

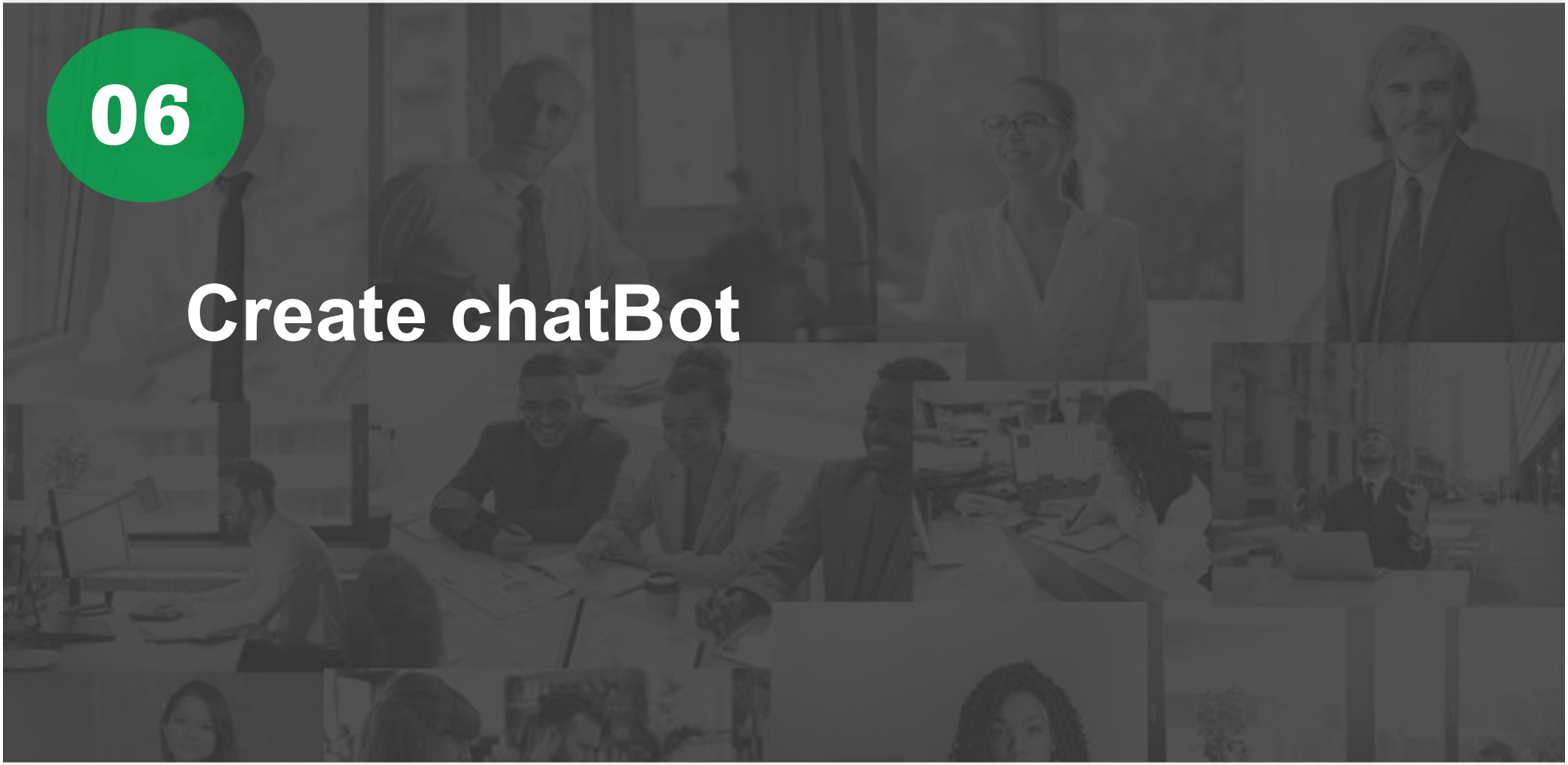
Get the information from the response object

```python
def FC_PROCESS_TASK(ZVFCI_AI_CLIENT, ZVFCI_AI_ASSISTANT, ZVFCI_AI_THREAD_ID, ZVFCI_ST_PR

    # 1/ Add thread, role and content to Client
    ZVFCI_AI_CLIENT.beta.threads.messages.create(
        thread_id=ZVFCI_AI_THREAD_ID,
        role="user",
        content=ZVFCI_ST_PROMPT
    )

    # 2/ Try to run AI query
    try:
        ZV_OB_OPENAI_RESPONSE = ZVFCI_AI_CLIENT.beta.threads.runs.create_and_poll(
            thread_id=ZVFCI_AI_THREAD_ID,
            assistant_id=ZVFCI_AI_ASSISTANT.id,
            tools=[]
        )

    except PI_AUHENTICAIONERROR as e:
        print("Authentication failed:", str(e))

    # 4.3/ Response
    # 4.3.1/ Default - no response
    ZV_ST_RESPONSE_MESSAGE = "No response found."

    # 4.3.2/ Get response if run of AI query successful
    if ZV_OB_OPENAI_RESPONSE:
        ZV_OB_LI_OPENAI_MSG = ZVFCI_AI_CLIENT.beta.threads.messages.list(thread_id=ZVFCI
        for ZV_OB_OPENAI_MSG in ZV_OB_LI_OPENAI_MSG.data:
            if ZV_OB_OPENAI_MSG.role == "assistant":
                content = ZV_OB_OPENAI_MSG.content[0]
                if hasattr(content, 'text'):
                    ZV_ST_RESPONSE_MESSAGE = content.text.value
                    break

                elif hasattr(content, 'image'):
                    ZV_ST_RESPONSE_MESSAGE = "[Image Response]"
                    break

    # 5/ Return output
    ZV_DI_AI_RESPONSE ={
        'ZV_ST_RESPONSE_MSG':ZV_ST_RESPONSE_MESSAGE,
    }

    return ZV_DI_AI_RESPONSE
```

Parse markdown

This part has been added today so that we can export the results to Excel (and not just print to the terminal)

```python
def FC_PARSE_MARKDOWN_TABLE(ZVFCI_ST_TABLE):

    ZV_LI_DI_PARSED_MARKDOWN_ROWS = []

    ZV_LI_ST_LINES = [
        ZV_ST_LINE.strip()
        for ZV_ST_LINE in ZVFCI_ST_TABLE.split('\n')
        if ZV_ST_LINE.strip().startswith('|')
    ]

    for ZV_ST_LINE in ZV_LI_ST_LINES:

        if '---' in ZV_ST_LINE:
            continue

        ZV_LI_ST_VALUES = [
            ZV_ST_VALUE.strip()
            for ZV_ST_VALUE in ZV_ST_LINE.strip('|').split('|')
        ]

        if ZV_LI_ST_VALUES == [
            'Issue',
            'Priority',
            'Observation',
            'Risk',
            'Recommendation'
        ]:
            continue

        if len(ZV_LI_ST_VALUES) == 5:

            ZV_LI_DI_PARSED_MARKDOWN_ROWS.append(
                {
                    'Issue': ZV_LI_ST_VALUES[0],
                    'Priority': ZV_LI_ST_VALUES[1],
                    'Observation': ZV_LI_ST_VALUES[2],
                    'Risk': ZV_LI_ST_VALUES[3],
                    'Recommendation': ZV_LI_ST_VALUES[4]
                }
            )

    return ZV_LI_DI_PARSED_MARKDOWN_ROWS
```



06

Create chatBot

Create chatBot

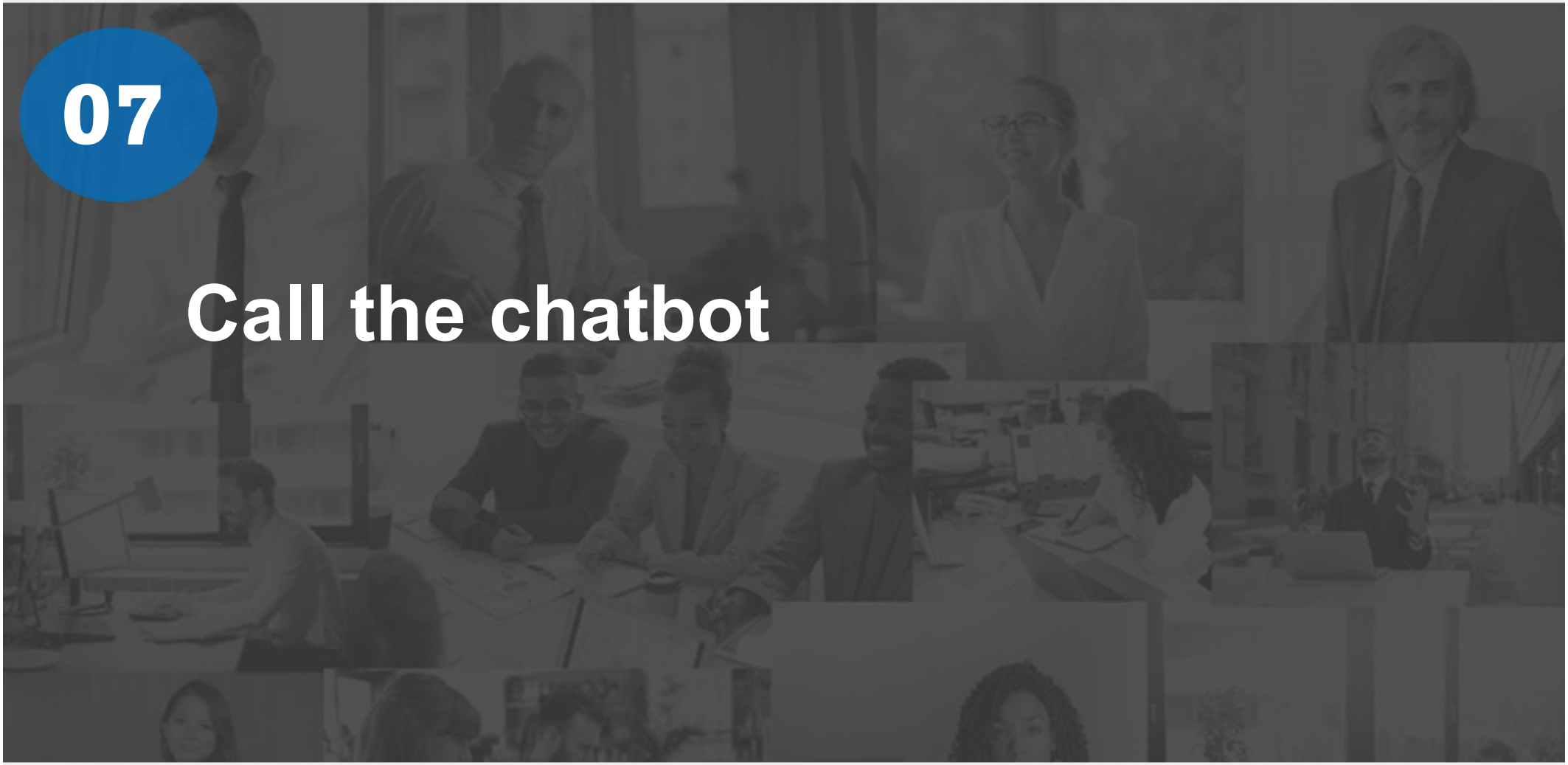
Here we print the results to terminal, but we could print to a file, as we do in 300FMC – we print to a PDF report.

```
237    def FC_CHATBOT():
238        # 1.1/ Create client and thread ID
239        ZV_OB_OPENAI_CLIENT, ZV_ST_THREAD_ID = FC_CREATE_AI_CLIENT()
240
241        # 1.2/ Create AI assistant
242        ZV_OB_ASSISTANT = FC_CREATE_AI_ASSISTANT(ZV_OB_OPENAI_CLIENT)
243
244        # 1.3/ Import JSON tasks
245        ZV_DI_ALL_TASKS = FC_LOAD_TASKS_FROM_JSON(ZV_ST_SOURCES_FOLDER, ZV_ST_JSON_FILE_NAME)
246
247        # 1.4/ Loop through all dashboards and tasks
248        ZV_LI_DI_RESPONSES = []
249
250        for ZV_ST_DASHBOARD_ID, ZV_LI_DASHBOARD_TASKS in ZV_DI_ALL_TASKS.items():
251            for ZV_IN_TASK_INDEX, ZV_DI_TASK in enumerate(ZV_LI_DASHBOARD_TASKS):
252                # print(f"\n{'='*80}")
253                # print(f"Processing: {ZV_ST_DASHBOARD_ID} - Task {ZV_IN_TASK_INDEX + 1}")
254                # print(f"{'='*80}\n")
255
256                # Create prompt
257                ZV_ST_PROMPT, ZV_ST_CHART_TITLE = FC_CREATE_AI_PROMPT(
258                    ZV_ST_SOURCES_FOLDER,
259                    ZV_ST_SOURCE_FILE_NAME,
260                    ZV_DI_TASK
261                )
262
263                # Process task
264                ZV_DI_OPENAI_RESPONSE = FC_PROCESS_TASK(ZV_OB_OPENAI_CLIENT, ZV_OB_ASSISTANT, ZV_S
265
266                # Prepare response
267                ZV_DI_OPENAI_RESPONSE = {
268                    'dashboard_id': ZV_ST_DASHBOARD_ID,
269                    'task_index': ZV_IN_TASK_INDEX + 1,
270                    'chart_title': ZV_ST_CHART_TITLE,
271                    'response': ZV_DI_OPENAI_RESPONSE['ZV_ST_RESPONSE_MSG'],
272                }
273
274                # Separate the table and Executive Summary if applicable
275                if "### Executive Summary" in ZV_DI_OPENAI_RESPONSE['response']:
276                    ZV_ST_RESPONSE_TABLE, ZV_ST_RESPONSE_SUMMARY = ZV_DI_OPENAI_RESPONSE['response
277                else:
278                    ZV_ST_RESPONSE_TABLE, ZV_ST_RESPONSE_SUMMARY = ZV_DI_OPENAI_RESPONSE['response
```

Create chatBot

To enable printing of the response, we parse the markdown information that is in in ZV_ST_TABLE and then we create a list of dictionaries, including this information.

```
313             # Parse the markdow information
314             ZV_LI_DI_PARSED_MARKDOWN_ROWS = FC_PARSE_MARKDOWN_TABLE(ZV_ST_RESPONSE_TABLE)
315
316             # Add to all responses
317             for ZV_DI_PARSED_MARKDOWN_ROW in ZV_LI_DI_PARSED_MARKDOWN_ROWS:
318
319                 ZV_LI_DI_RESPONSES.append(
320                     {
321                         'dashboard_id': ZV_ST_DASHBOARD_ID,
322                         'task_index': ZV_IN_TASK_INDEX + 1,
323                         'chart_title': ZV_ST_CHART_TITLE,
324                         'Issue': ZV_DI_PARSED_MARKDOWN_ROW['Issue'],
325                         'Priority': ZV_DI_PARSED_MARKDOWN_ROW['Priority'],
326                         'Observation': ZV_DI_PARSED_MARKDOWN_ROW['Observation'],
327                         'Risk': ZV_DI_PARSED_MARKDOWN_ROW['Risk'],
328                         'Recommendation': ZV_DI_PARSED_MARKDOWN_ROW['Recommendation'],
329                         'Executive Summary': ZV_ST_RESPONSE_SUMMARY.strip()
330                     }
331                 )
332
333         # Return all responses
334         return ZV_LI_DI_RESPONSES
335
```


07
Call the chatbot

Call the chatbot – just press play!

Now when we call FC_CHATBOT, we obtain a list of dictionaries that are our responses.

This list of dictionaries can be converted to a DataFrame and then exported to Excel.

```
334
335
336     ZV_LI_DI_RESPONSES = FC_CHATBOT()
337
338     ZV_DF_AI_RESPONSES = PI_POLARS.DataFrame(ZV_LI_DI_RESPONSES)
339
340     FC_EXPORT_EXCEL_POLARS(
341         ZV_DF_AI_RESPONSES,
342         ZV_ST_RESULTS_FOLDER,
343         ZV_ST_RESULTS_FILE_NAME
344     )
345
346 if __name__ == '__main__':
347     Z00_OPENAI_API()
348
```

Call the chatbot – result!

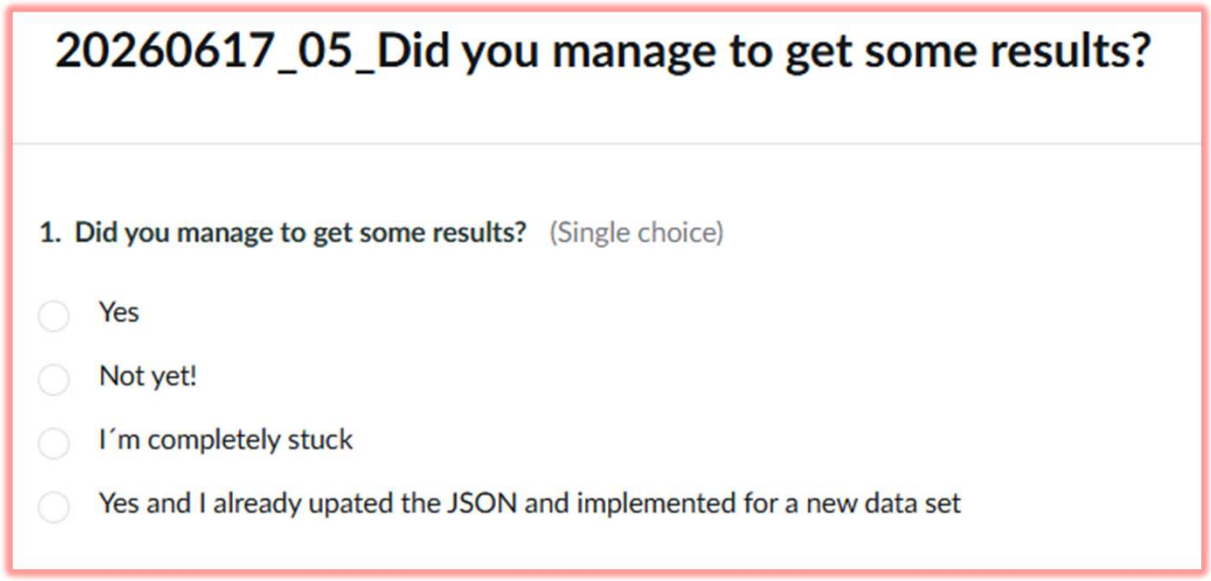
Our result – information printed to screen... and exported!

TERMINAL

```
================================================================================
  CHART 1/ TOTAL VALUES PER MONTH
================================================================================

### Observations
--------------------------------------------------------------------------------
The total values per month show a generally increasing trend, with February marking the highest value of $150,000, followed by May at $160,000. Howe
ver, March and April indicate a slight decline in values compared to February.


### Risk Assessment
--------------------------------------------------------------------------------
| Issue                 | Priority | Observation                    | Risk                           | Recommendation                      |
|-----------------------|----------|--------------------------------|--------------------------------|-------------------------------------|
| Month-to-Month Decline| Medium   | March and April saw a decline  | Potential sales opportunity loss | Investigate reasons for decline     |
| Seasonal Variability  | Medium   | Fluctuations in monthly values | Inaccurate forecasting         | Perform seasonality analysis        |

--------------------------------------------------------------------------------
  EXECUTIVE SUMMARY
--------------------------------------------------------------------------------
The analysis of the total values per month reveals a strong upward trend overall, with a notable peak in February and May. However, the decline obse
rved in March and April raises concerns about potential sales opportunities being missed. Additionally, the fluctuation in monthly values may indica
te issues with forecasting accuracy. It is recommended to investigate the causes of the decline and to conduct a seasonality analysis to improve fut
ure predictions and strategy alignment.

================================================================================
```

Did you manage to get some results?

Did you manage to get some results?



20260617_05_Did you manage to get some results?

1. Did you manage to get some results? (Single choice)

- Yes
- Not yet!
- I´m completely stuck
- Yes and I already upated the JSON and implemented for a new data set

Questions?