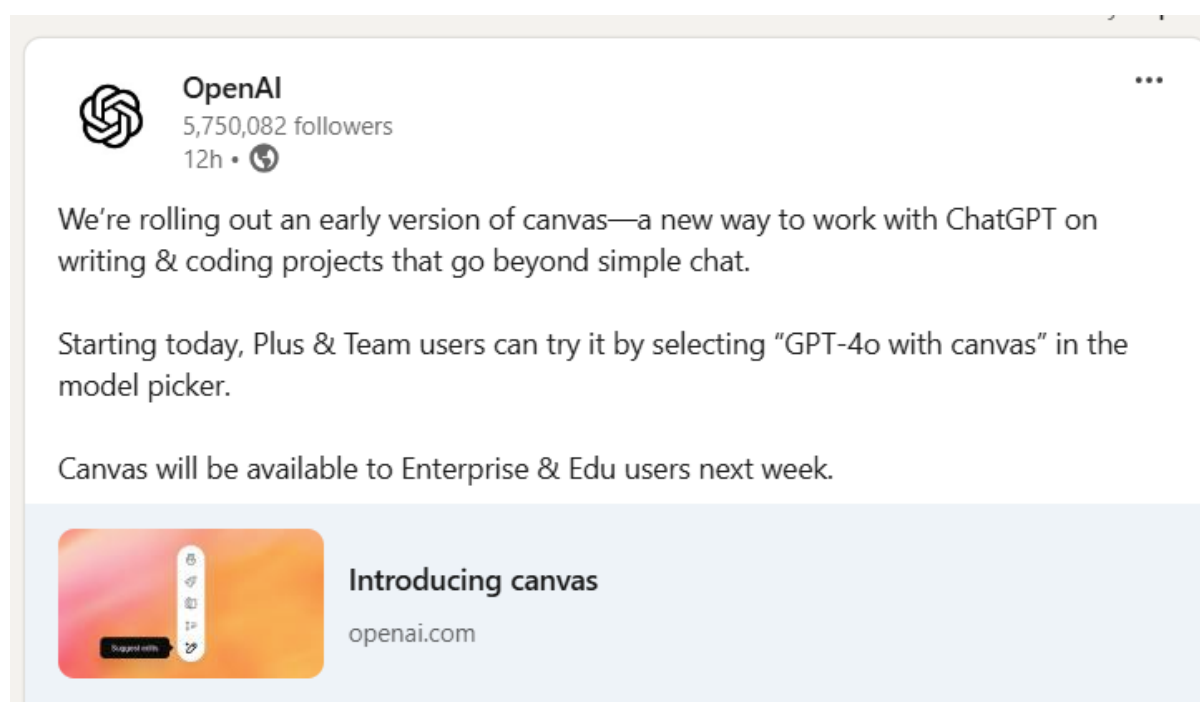


# ChatGPT Canvas for FP&A and Finance

OpenAI has released canvas, a new interface for working with ChatGPT on writing and coding projects that go beyond simple chat.

Canvas opens in a separate window, allowing you and ChatGPT to collaborate on a project. This early beta introduces a new way of working together—not just through conversation, but by creating and refining ideas side by side.



I will show you in this guide how you can leverage it for FP&A and Finance.

## What Is ChatGPT Canvas?

Canvas is a newly introduced interactive interface for ChatGPT that allows users to work visually on various projects.

Designed for tasks that require deeper contextual understanding—like financial modeling, scenario analysis, and strategic planning—Canvas provides a shared

space for editing, refining, and iterating on complex data sets and financial narratives.

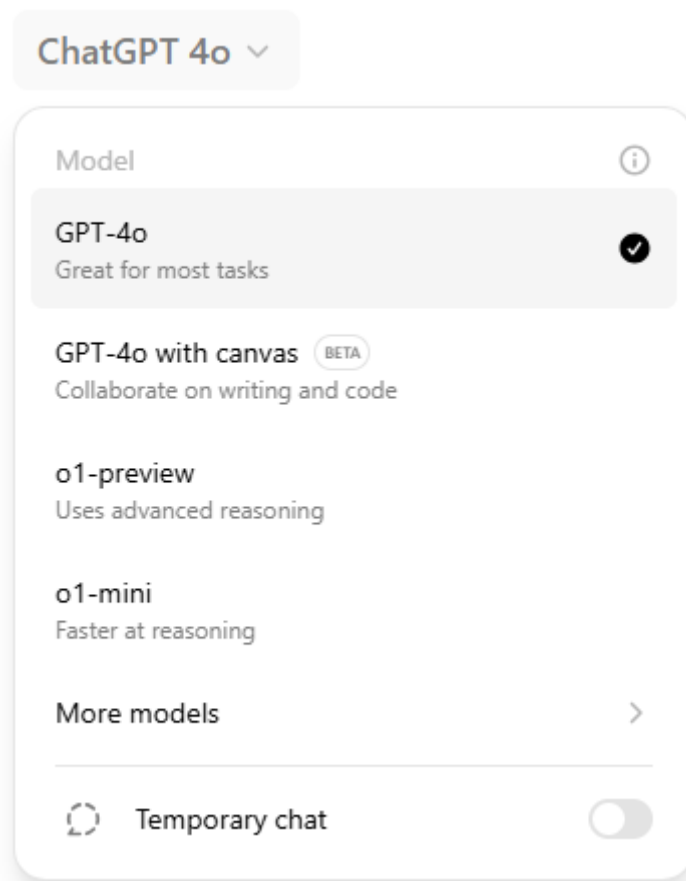
Canvas supports markdown formatting, version control, and direct in-line feedback, making it ideal for producing and reviewing financial documents.

### **How to enable Canvas and who gets access?**

Canvas was built with GPT-4o and can be manually selected in the model picker while in beta.

Starting yesterday 3th October 2024, OpenAI is rolling out canvas to ChatGPT Plus and Team users globally.

Enterprise and Edu users will get access the following week. They also plan to make canvas available to all ChatGPT Free users when it's out of beta.



# Use Cases for ChatGPT Canvas in FP&A and Finance

## 1) Financial Modelling and Forecasting

This is one of my favourites, it can help with financial modelling, let's say that you are trying to use Python to create a financial forecast.

You can try with this prompt:

*"I want to use Python to create a financial forecast with Prophet. Generate Python code for me to do it. I will be using Google Colab and my data looks like this:*

*\*\*add data here\*\* "*

ChatGPT 4o with canvas



Create a workout plan



Experience Seoul like a local



Fun fact about the Roman Empire



What to do with kids' art

I want to use Python to create a financial forecast with Prophet. Generate Python code for me to do it. I will be using Google Colab and my data looks like this:

	Sales Subcategory	product	product ID	Sales Value in Units	Qty	Sales Value in USD
Sales Unit Price						
Distributor	Router	P00015	607	108517.3981	178.7766032	
Direct	Game Console	P00004	277	193850.8134	699.822431	

It will give you the code which you can co-edit with it.

```
Prophet (ds: column) and the target sales value (y: column) is specified.

4. Model Initialization and Forecasting: The Prophet model is initialized and fit on the data. We then create future dates and predict sales for the next 12 months.

5. Visualization: The forecast is plotted, and we also print out the forecasted values.

Make sure to adjust the column names in the code to match those in your dataset. Prophet requires a 'ds' (datetime) and 'y' (target variable) column, so the key step is to convert your existing data accordingly.

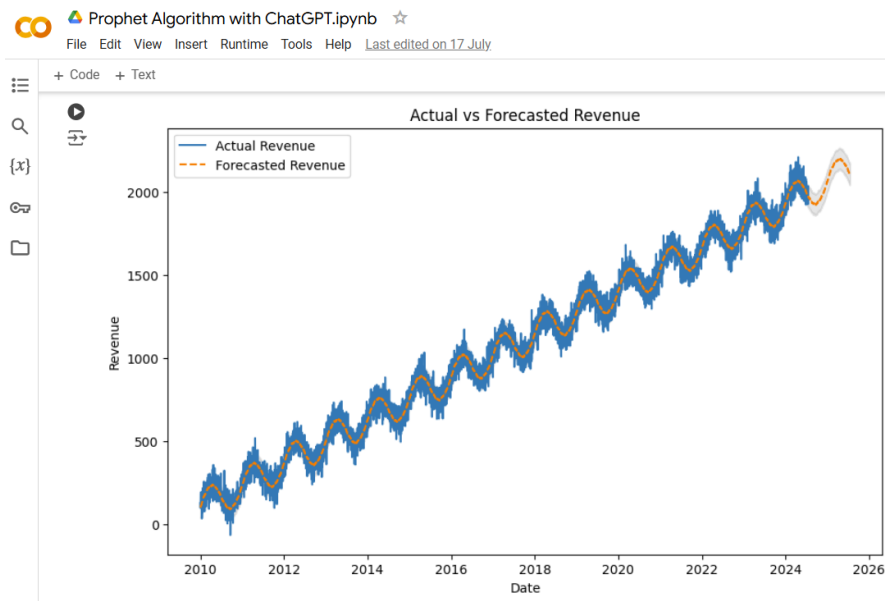
If you need any further modifications, such as changing the frequency or adding additional components like holidays, let me know!
```

```
Prophet Sales Forecast

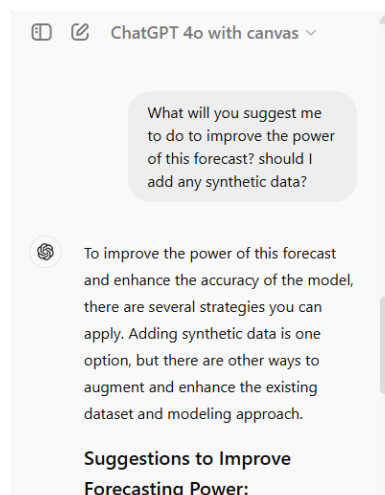
8
9
10 # Assuming you have uploaded your CSV file to Google Colab, you can load it as follows
11 # Load the dataset (replace 'your_data.csv' with the name of your uploaded file)
12 data = pd.read_csv('your_data.csv')
13
14 # Display the first few rows to understand the structure of your data
15 data.head()
16
17 # Preprocess the data to make it suitable for Prophet
18 # Prophet requires a dataframe with 'ds' and 'y' columns:
19 # 'ds' is the date/time column, and 'y' is the target value we want to forecast
20
21 # For the purpose of this forecast, let's assume your sales data is recorded on a monthly basis.
22 # If you have a 'date' column, we need to parse that column.
23 # Suppose you have a Date column named 'Month-Year' (e.g., 'Jan-2024'):
24
25 # Convert 'Month-Year' to datetime (change column name as appropriate)
26 data['date'] = pd.to_datetime(data['Month-Year'], format='%b-%Y')
27
28 # Create a new dataframe with the 'ds' and 'y' columns for Prophet
29 sales_forecast_data = pd.DataFrame()
30 sales_forecast_data['ds'] = data['date']
31 sales_forecast_data['y'] = data['Sales Value in USD']
32
33 # Initialize and fit the Prophet model
34 model = Prophet()
35 model.fit(sales_forecast_data)
36
37 # Create future dates to forecast (for example, forecast for the next 12 months)
38 future = model.make_future_dataframe(periods=12, freq='M')
39
40 # Predict future sales
41 forecast = model.predict(future)
42
43 # Plot the forecast
44 fig = model.plot(forecast)
45 plt.title("Sales Forecast using Prophet")
46 plt.xlabel("Time")
47 plt.ylabel("Sales Value in USD")
48 plt.show()
49
50 # Display the forecasted data
51 forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(12) # show the last 12 months of forecast
```

Then, similarly to what I explained on my previous guides, you will go to Google Colab, Python in Excel, Visual Studio or another platform and run the code with your data.

You can ask it to forecast any variable like revenue:



And you can ask as well how to improve your forecast or even use the shortcuts (more on that on the last pages).



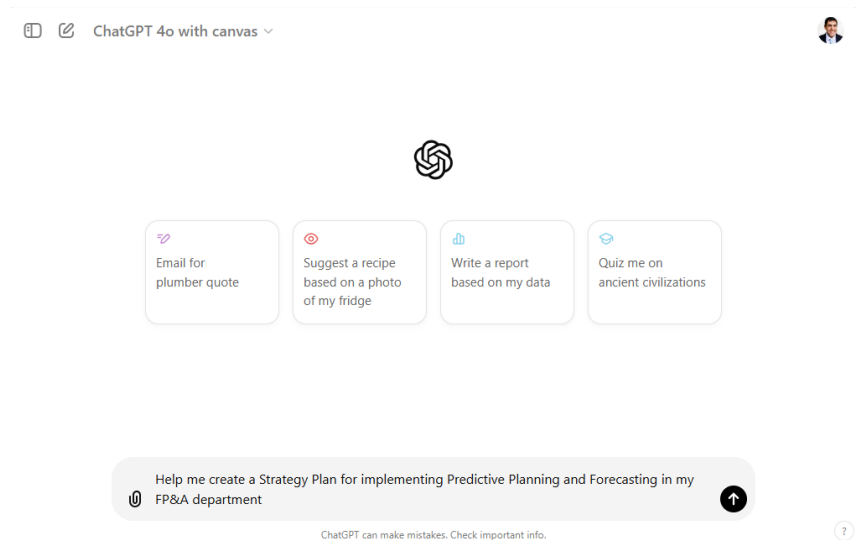
generate synthetic data for this

```
29 sales_forecast_data = pd.DataFrame()
30 sales_forecast_data['ds'] = data['Date']
31 sales_forecast_data['y'] = data['Sales Value in USD']
32
33 # Generate synthetic data to enhance the power of the f
34 # Assume the sales values follow some trend and seasona
35 np.random.seed(42) # For reproducibility
36
37 # Create additional synthetic data points for the past
38 additional_dates = pd.date_range(start="2018-01-01", en
39
40 # Predict future sales
41 forecast = model.predict(future)
42
43 # Plot the forecast
```

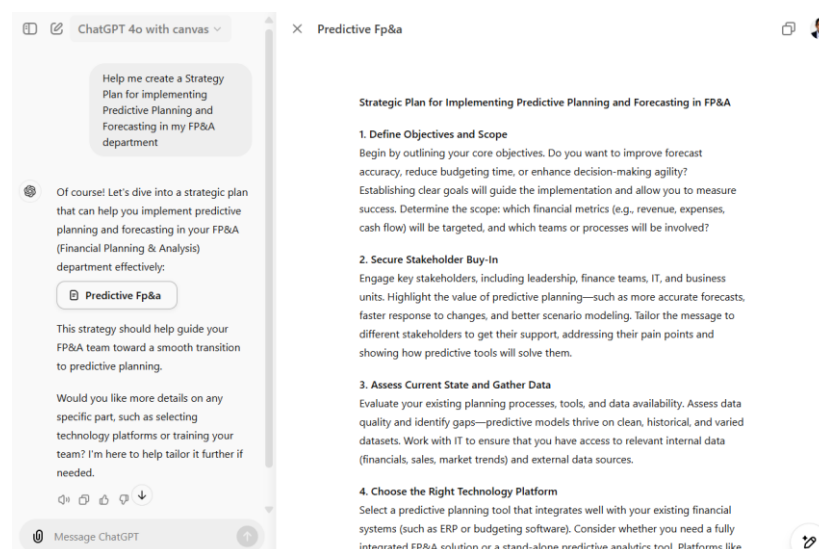
## 2) Strategy Planning for FP&A

For example, you can try with this prompt:

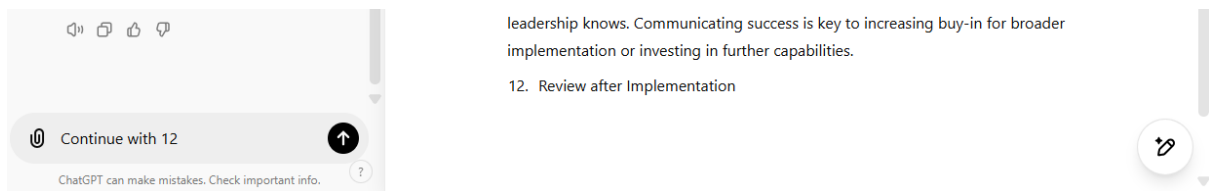
*"Help me create a Strategy Plan for implementing Predictive Planning and Forecasting in my FP&A department"*



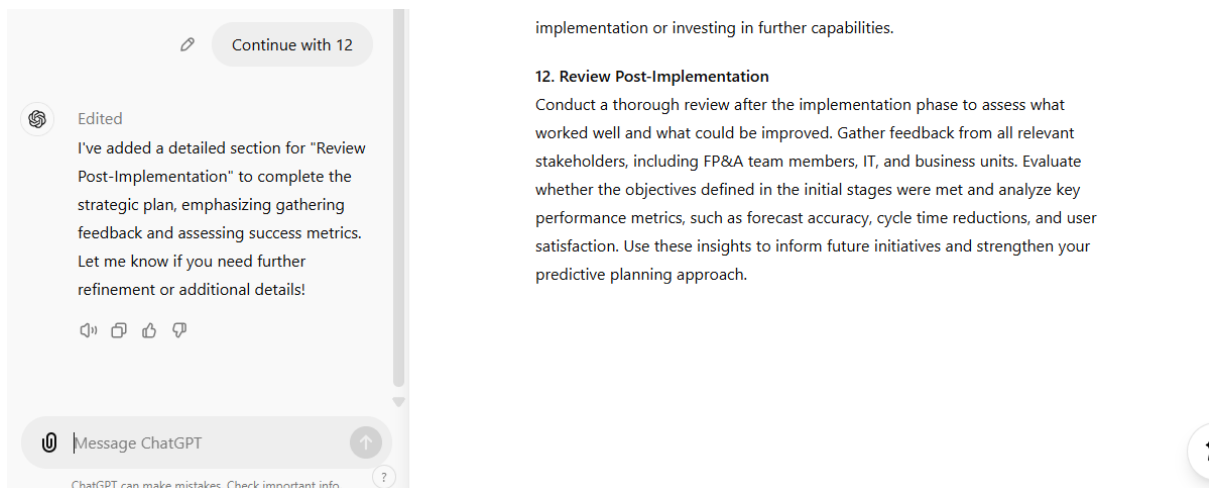
ChatGPT 4o with canvas will start writing:



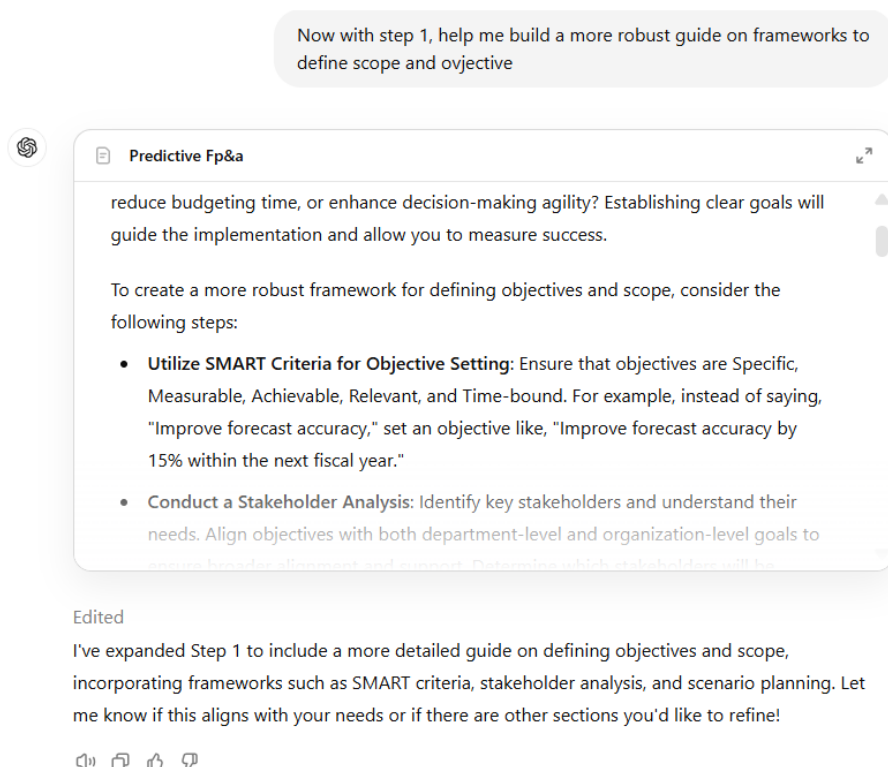
You can now edit the output of ChatGPT and suggest changes, for example add a step:



I told it what I wanted but it wrote the full paragraph:



At first, it will be very broad but you can prompt it to go deeper in any of the steps you might need more practical advices, like this:



### 3) Data Visualization and Analysis

For example, you might want to do a cohort analysis which is a method used to group customers into categories, or "cohorts," based on shared characteristics, often related to the time they performed a certain action (e.g., their first purchase).

In SaaS, these characteristics could be based on the month they signed up for your service.

You can try with this prompt:

*"I want to get Python code to create a Cohort Analysis for a SaaS company, my data looks like this and is in an Excel file called data. I will be using Google Colab:*

*\*\*add your data here\*\* "*

It will give you code for it:

The screenshot shows a ChatGPT interface with a prompt on the left and a code response on the right. The prompt asks for Python code to create a cohort analysis for a SaaS company, with data in an Excel file named 'data'. The data is shown as a table with columns: Date, Customer\_ID, Product, Invoice, and Invoice Amount. The response is a Python script that loads the data, parses the date, extracts the cohort group based on the month of the first purchase, calculates the cohort index, and generates a heatmap of customer retention rates.

**Prompt:**

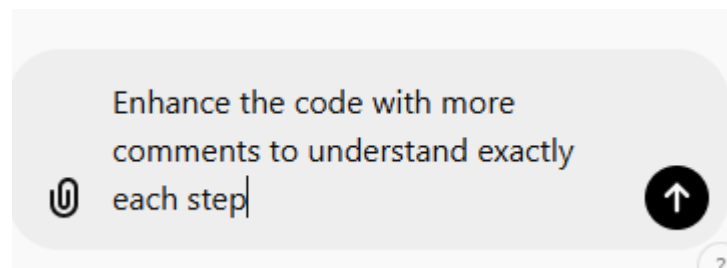
I want to get Python code to create a Cohort Analysis for a SaaS company, my data looks like this and is in an Excel file called data. I will be using Google Colab:

Date	Customer_ID	Product	Invoice	Invoice Amount
31/01/2022	1	Basic	1000	5000
31/01/2022	2	Enterprise	5000	3000
31/01/2022	3	Pro	3000	5000
31/01/2022	4	Pro	3000	5000
31/01/2022	5	Enterprise	5000	1000
31/01/2022	6	Basic	1000	

**Response:**

```
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from google.colab import files
6
7 # Upload the data from your local system
8 uploaded = files.upload()
9
10 # Load the Excel file into a DataFrame
11 data = pd.read_excel('data.xlsx')
12
13 # Parse date column to datetime format
14 data['Date'] = pd.to_datetime(data['Date'], dayfirst=True)
15
16 # Extract the cohort group based on the month of each customer's first purchase
17 data['CohortGroup'] = data.groupby('Customer_ID')['Date'].transform('min').dt.to_period('M')
18
19 # Extract the order month and cohort month
20 order_month = data['Date'].dt.to_period('M')
21 cohort_month = data['CohortGroup']
22
23 # Calculate the cohort index (the difference in months between each order month and the cohort month)
24 cohort_index = (order_month - cohort_month).apply(lambda x: x.n)
25
26 # Create a cohort analysis pivot table
27 cohort_data = data.pivot_table(
28     index='CohortGroup',
29     columns=cohort_index,
30     values='Customer_ID',
31     aggfunc=pd.Series.nunique
32 )
33
34 # Divide by the cohort size (number of customers in each cohort) to get retention rates
35 cohort_size = cohort_data.iloc[:, 0]
36 retention_matrix = cohort_data.divide(cohort_size, axis=0)
37
38 # Plotting the retention matrix using a heatmap
39 plt.figure(figsize=(12, 8))
40 sns.heatmap(retention_matrix, annot=True, fmt='.0%', cmap='Blues')
41 plt.title('Cohort Analysis - Customer Retention Rate')
42 plt.xlabel('Cohort Index')
43 plt.ylabel('Cohort Group')
44 plt.show()
```

You can also either ask to enhance the code and get more comments so you can understand each part of the code or even use the shortcuts (more on that on the last pages).

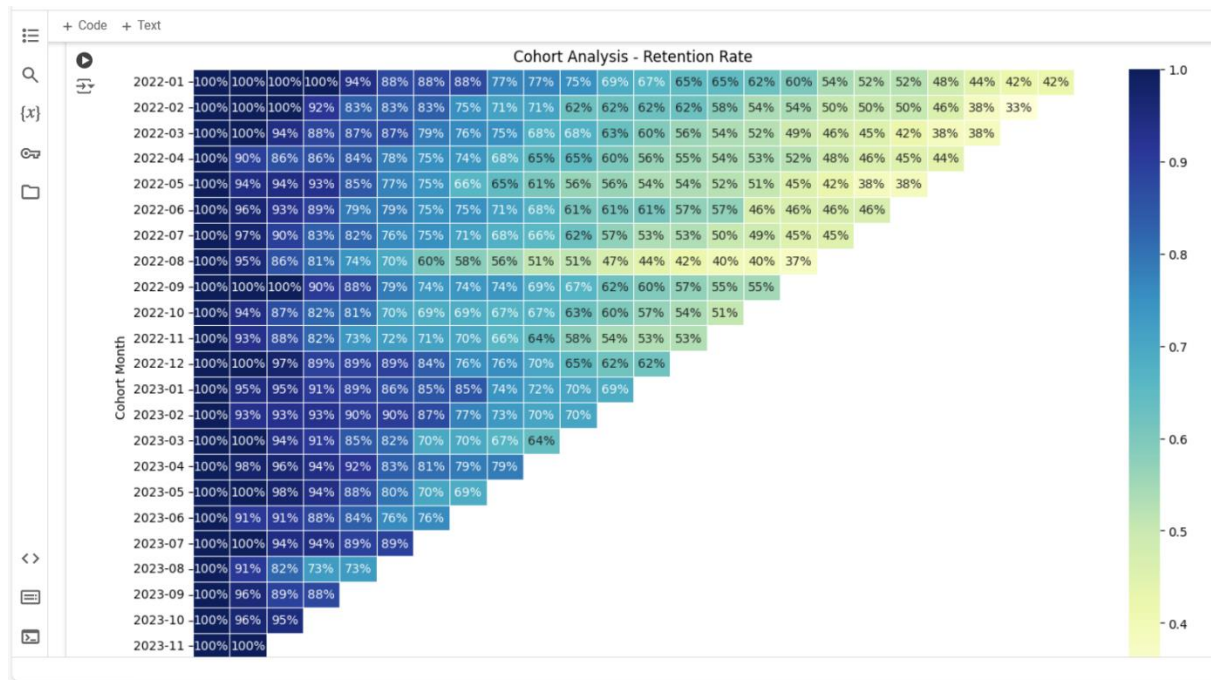


#### ✕ Cohort Analysis Saas

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import seaborn as sns
5  from google.colab import files
6
7  # Upload the data from your local system
8  # This will prompt you to upload an Excel file containing the data
9  uploaded = files.upload()
10
11 # Load the Excel file into a DataFrame
12 # Replace 'data.xlsx' with the name of the uploaded file if different
13 data = pd.read_excel('data.xlsx')
14
15 # Parse the 'Date' column to datetime format
16 # This ensures that the dates are properly recognized and can be used for time-based operations
17 data['Date'] = pd.to_datetime(data['Date'], dayfirst=True)
18
19 # Extract the cohort group based on the month of each customer's first purchase
20 # 'CohortGroup' represents the month when each customer made their first purchase
21 data['CohortGroup'] = data.groupby('Customer_ID')['Date'].transform('min').dt.to_period('M')
22
23 # Extract the order month and cohort month
24 # 'order_month' represents the month of each purchase
25 # 'cohort_month' represents the cohort (first purchase month) for each customer
26 order_month = data['Date'].dt.to_period('M')
27 cohort_month = data['CohortGroup']
28
29 # Calculate the cohort index (the difference in months between each order month and the cohort month)
30 # This helps determine how many months have passed since the customer's first purchase
31 cohort_index = (order_month - cohort_month).apply(lambda x: x.n)
32
```



This will be the output of a cohort analysis in Google Colab:



## ChatGPT 4o with Canvas Shortcuts

Developing financial models and analysis tools is an iterative process, and tracking each change within a standard chat interface can quickly become overwhelming.

Canvas offers a streamlined way to monitor and review the revisions made to your code, making it easier to manage complex financial projects.

You can prompt it like I did in the examples above or use the shortcuts.

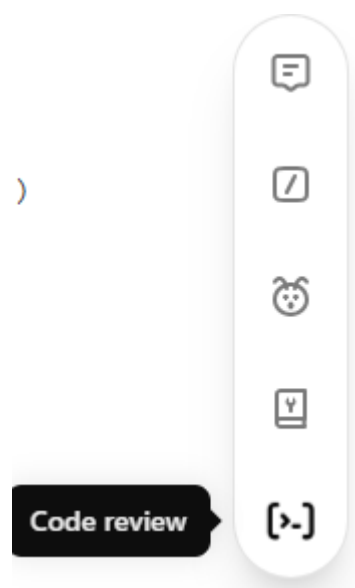
These **shortcuts** are quick-access commands or actions designed to simplify and accelerate specific tasks.

They enable users to perform repetitive or complex operations with a single click or command, thereby reducing the amount of manual editing or coding needed.

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from google.colab import files
6
7 # Upload the data from your local system
8 # This will prompt you to upload an Excel file containing the data
9 uploaded = files.upload()
10
11 # Load the Excel file into a DataFrame
12 # Replace 'data.xlsx' with the name of the uploaded file if different
13 data = pd.read_excel('data.xlsx')
14
15 # Parse the 'Date' column to datetime format
16 # This ensures that the dates are properly recognized and can be used for time-based operations
17 data['Date'] = pd.to_datetime(data['Date'], dayfirst=True)
18
19 # Extract the cohort group based on the month of each customer's first purchase
20 # 'CohortGroup' represents the month when each customer made their first purchase
21 data['CohortGroup'] = data.groupby('Customer_ID')['Date'].transform('min').dt.to_period('M')
22
23 # Extract the order month and cohort month
24 # 'order_month' represents the month of each purchase
25 # 'cohort_month' represents the cohort (first purchase month) for each customer
26 order_month = data['Date'].dt.to_period('M')
27 cohort_month = data['CohortGroup']
28
29 # Calculate the cohort index (the difference in months between each order month and the cohort month)
30 # This helps determine how many months have passed since the customer's first purchase
31 cohort_index = (order_month - cohort_month).apply(lambda x: x.n)
```

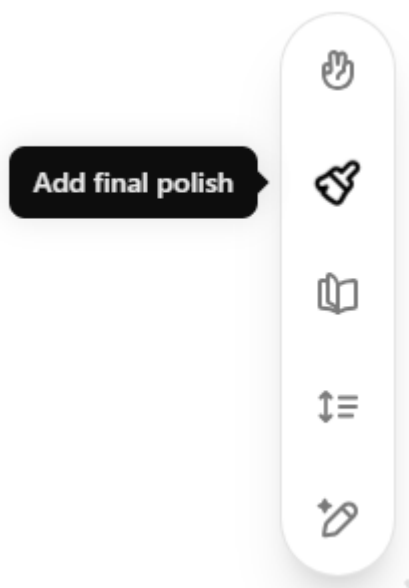
## Coding Shortcuts for FP&A:

- **Review Code:** Provides targeted suggestions to optimize and improve your Python, VBA, or SQL scripts.
- **Add Comments:** Generates detailed comments in your code, explaining functions and calculations for better clarity and documentation.
- **Fix Errors:** Identifies and resolves coding errors, ensuring your financial scripts run smoothly and accurately.
- **Convert to a Programming Language:** Converts your code into other finance-relevant languages, such as switching between Python, VBA, or SQL, to enhance compatibility with different financial systems.



## Writing Shortcuts for FP&A Reports:

- **Suggest Edits:** Provides in-line recommendations to refine and enhance your financial narratives.
- **Adjust the Length:** Modifies the text to be more concise or expands it for deeper detail, depending on your needs.
- **Change Reading Level:** Adapts the language complexity, from simple explanations to advanced business language.
- **Add Final Polish:** Reviews for grammar, clarity, and consistency to ensure a professional tone.
- **Add Emojis:** Adds context-appropriate emojis to highlight key points or add visual emphasis.



## Model Evaluations

OpenAI evaluated the model and got these results:

