eFuturesCFO Masterclass Series

AI Workflows for the Modern CFO

PART 1

AI Foundations for the Finance Executive

The Concepts Every CFO Must Master Before Touching the Tools

Hindol Datta

Chief Content and Product Officer, eFuturesCFO

A Note Before You Begin

This masterclass exists because the relationship between artificial intelligence and the office of the chief financial officer is now one of the defining executive questions of the decade. A CFO who does not understand this technology will be governed by it. A CFO who does understand it will govern it. There is no third position available, and the window for choosing is shorter than most finance leaders currently believe.

I have written this masterclass for the working CFO, the controller preparing to become a CFO, the finance leader managing an AI initiative inside a larger organization, and the board member responsible for overseeing finance in a company where AI is being deployed. The language is plain. The analogies are simple. The depth is real.

Part 1, which you are about to read, is the foundation. It establishes the vocabulary, the architecture, and the conceptual model that every subsequent part of the masterclass assumes. If you read nothing else in this series, read this part carefully. The use cases in Parts 5 through 9 will mean very little if the foundation is shaky, and they will be transformative if the foundation is solid.

I have deliberately avoided two failure modes that afflict most executive AI education. The first is the failure of unwarranted sophistication, where the writer demonstrates fluency in technical terminology without ever explaining what the terminology means. The second is the failure of unwarranted simplification, where the writer reduces the technology to a set of cheerful slogans that fall apart the moment the executive encounters reality. Neither serves the reader. The right standard is the standard a serious eighth-grade teacher would apply: explain it so that any thoughtful person can understand it, but explain it accurately enough that the understanding is real.

Throughout this part you will encounter twelve sections, each addressing one major concept. You will also encounter callout boxes that contain the analogies I rely on most heavily. At the end of the part there is a glossary that consolidates every term introduced in the foundation, and a twenty-question assessment that will test whether the foundation is actually solid before you move forward.

Read it slowly. Underline what surprises you. Argue with the parts that feel wrong. The point is not to finish the part. The point is to acquire a working mental model of how this technology operates, what it can do, what it cannot do, and where the executive authority of the CFO most usefully attaches.

> *A CFO who does not understand this technology will be governed by it. A CFO who does understand it will govern it.*

Hindol Datta

Fremont, California

Contents

The page numbering begins with the first section. Use the running header at the top of each page to orient yourself within the part.

Section 1 · What an AI Model Actually Is

The single most important sentence in the foundation

A large language model is a very large pattern-recognition engine that has been trained on a very large quantity of human text. It does not know things in the way a person knows things. It predicts the next likely word, one word at a time, until a response is complete.

That sentence is the entire foundation. Every behavior you will encounter when working with ChatGPT, Claude, or any other AI assistant follows from that sentence. The fluency follows from it. The hallucination follows from it. The occasional brilliance follows from it. The occasional embarrassment follows from it. A CFO who internalizes this single sentence has solved sixty percent of the difficulty of working with this technology.

> **The library and the apprentice**
>
> Imagine an exceptionally diligent apprentice who has spent ten years in the largest library on Earth, reading every book, every article, every conversation, every piece of code, every transcript, every research paper. The apprentice does not remember any specific book. The apprentice remembers patterns. When you ask a question, the apprentice does not look anything up. The apprentice predicts what a knowledgeable answer would sound like, based on every pattern of language and reasoning observed across ten years of reading. This is what a large language model is. It is a pattern-predicting apprentice, not a librarian.

Why this matters for finance

A finance executive who treats the AI as a librarian will be repeatedly disappointed. The model does not retrieve facts from a database. It generates text that resembles the kind of text its training would predict. When the prediction is well-calibrated, the output is accurate. When the prediction is poorly calibrated, the output is wrong but still confident, because the model has no mechanism for representing uncertainty in the way a human does.

This is the origin of what is called hallucination. The model does not lie. It does not even know that it is wrong. It is generating a continuation that fits the pattern of a confident answer, even when the underlying knowledge is absent or distorted. A CFO who understands this never trusts the model unconditionally. The CFO designs workflows that verify, that constrain, that ground the model in real data, and that never allow the model to substitute for the chain of evidence that finance requires.

The training process in plain language

The technical details of how a large language model is trained are beyond the scope of this masterclass and beyond the scope of what a CFO needs to know. The high-level mechanism is simple. The model is shown enormous quantities of text and asked, again and again, to predict the next word. Each time it predicts correctly, the internal parameters are reinforced. Each time it predicts incorrectly, the parameters are adjusted. After this process is run trillions of times across hundreds of billions of words, the model develops a remarkable capacity to predict not just the next word but the next paragraph, the next argument, the next chain of reasoning.

What emerges from this process is something genuinely surprising to its own creators. The model develops capacities that no one explicitly programmed into it. It can summarize. It can translate. It can write code. It can construct arguments. It can analyze financial data. These capacities are not features that were added in a roadmap. They are emergent properties of pattern recognition at sufficient scale. The researchers who built these systems did not predict most of what the systems can do. This is a deeply important fact, and one that should make any thoughtful executive humble in their assertions about what the technology will or will not be able to do six months from now.

What the model does not have

Equally important is what the model does not have. The model has no senses. It cannot see the world unless an image is given to it. It cannot hear the world unless audio is transcribed for it. It has no memory across conversations unless that memory is deliberately constructed and provided to it. It has no ability to verify its own output against external reality. It cannot take an action in the world unless given a tool with which to act. It has no preferences, no fears, no desires, no internal monologue between exchanges, and no continuous existence.

Each conversation begins with the model essentially blank, except for whatever has been written into the immediate prompt and whatever instructions the developer has placed into what is called the system message. The model is, in a meaningful sense, a fresh instance every time you open a new chat. There is no AI somewhere in a data center thinking about your last conversation. There is a model that, when invoked, produces text in response to text. That is the entire mechanism.

> **The implication for finance leaders**
>
> Because the model has no memory and no persistent existence, any system that needs to remember things across conversations must be designed to remember on the model's behalf. This is why the masterclass spends so much time on architecture. The intelligence is in the model. The memory, the data, the governance, the audit trail, the workflow logic — all of these must be built around the model by people who understand what the model can and cannot do.

The difference between knowing and predicting

A human accountant who has been working in revenue recognition for twenty years knows the rules. They can cite the standard. They can point to the paragraph. They have a model of why the rules exist, who wrote them, what corner cases they were meant to address. When the accountant gives you an answer about revenue recognition, the answer is anchored in something verifiable.

A large language model that has been trained on revenue recognition literature can produce an answer that sounds identical to the accountant's answer, and frequently the answer will be correct. But the model is not anchored. The model is predicting what a correct-sounding answer would be. When the question lies near the center of well-documented territory, the prediction is highly reliable. When the question lies at the edge, in a domain the model has seen less of, the prediction begins to drift. The drift is often invisible, because the prose remains fluent and confident even as the content becomes unreliable.

This is why grounding matters. Grounding is the practice of giving the model the specific source material it needs to answer a question, rather than relying on the model's training. A model asked to summarize the revenue recognition policy in a company's audit committee memo will produce a far more reliable answer if the memo itself is included in the prompt than if the model is asked to generate the summary from its general training. This is the entire premise of retrieval-augmented generation, which we will return to in Section 9.

Section 2 · Tokens and the Economics of Words

The unit of measurement

Every interaction with a large language model is measured in tokens. A token is a small chunk of text, typically representing a piece of a word, an entire short word, a punctuation mark, or a space. On average, one token corresponds to roughly three-quarters of an English word, or about four characters. The word "finance" is one token. The word "extraordinarily" is three or four tokens. A sentence of fifteen English words is typically around twenty tokens. A page of text is roughly five hundred tokens. A short novel is roughly one hundred thousand tokens.

> **Why tokens exist**
>
> Computers process language by breaking it into small pieces. These pieces are not always whole words, because many words share components — prefixes, suffixes, roots — that the model can recognize and recombine. The tokenizer, which is the software that breaks text into tokens, is a piece of engineering that has been optimized for efficiency. You do not need to think about how it works. You need to think about what it means for cost, speed, and capacity.

Why tokens matter for the CFO

Tokens matter to the CFO because tokens are the currency of the AI economy. Every commercial AI provider prices their service in tokens. Every API call, every chat message, every document analysis, every agent action is measured in tokens. The reason a finance leader must understand this is that the gap between what a vendor quotes and what an enterprise actually spends is often enormous, and the gap is almost always denominated in tokens that the buyer did not anticipate.

Consider a concrete example. A vendor offers an AI-powered document analysis service at a quoted rate that sounds attractive. When the finance team begins using the service at scale, the bill is many multiples of the quoted rate. The reason is that each document analysis includes not only the document itself, but also the system prompt that configures the model, the prior conversation history if any, the metadata, the formatting instructions, and the output. Each of these consumes tokens. The vendor's quoted rate was technically accurate. The vendor's quoted rate was also, in practice, deeply misleading. A finance leader who understands tokens asks the right questions at the contract stage.

Input tokens and output tokens

Tokens come in two varieties: input tokens and output tokens. Input tokens are the tokens you send to the model. They include your question, any context you provide, the system prompt that configures the model's behavior, and any conversation history. Output tokens are the tokens the model generates in response. Output tokens are typically priced three to five times higher than input tokens, because generating text is computationally more expensive than reading it.

This asymmetry has practical implications for workflow design. A workflow that asks the model to read a large document and produce a short summary is cost-efficient. A workflow that asks the model to read a short prompt and produce a long elaboration is more expensive per word of useful output. The CFO who designs workflows with this asymmetry in mind can produce more value at lower cost. The CFO who ignores it will be surprised by the monthly bill.

> **A rule of thumb for cost estimation**
>
> For the major commercial models in mid-2026, a useful mental shortcut is that one million input tokens costs between three dollars and fifteen dollars depending on the model, and one million output tokens costs between fifteen dollars and seventy-five dollars. One million tokens is roughly seven hundred fifty thousand words, or about fifteen hundred pages. For most finance workflows, the cost per useful output is measured in cents or fractions of a cent. The cost becomes meaningful at scale, when thousands of workflows run per day across an enterprise.

Tokens and price differentiation across models

Different models have different token prices. The most capable models are typically the most expensive. The smaller and faster models are cheaper but produce less sophisticated output. A CFO designing an AI architecture must make deliberate choices about which model to use for which task. A board commentary generator probably justifies the cost of the most capable model. An expense categorization workflow probably does not. The art of finance AI architecture is, in significant part, the art of matching the task to the appropriate model tier.

The model tiers also evolve. The most capable model of today is the mid-tier model of next year, at a fraction of the cost. The CFO should not lock the architecture into a specific model. The architecture should be designed for model substitutability. We will return to this principle in Part 4 on governance.

Estimating tokens for your own workflows

A useful exercise for any CFO is to estimate the token consumption of a planned workflow before approving it. The estimation is simple. Count the words in a typical input. Multiply by 1.3 to convert words to tokens. Add the system prompt tokens, which are typically a few hundred to a few thousand depending on complexity. Estimate the output tokens by considering how long the response should be. Multiply input tokens by the input price and output tokens by the output price. Multiply by the number of invocations per month. The result is the monthly cost of the workflow.

This calculation, done before deployment, prevents the most common budget surprise in enterprise AI. It also forces the team designing the workflow to think explicitly about cost, which tends to produce more efficient designs. The CFO who institutes this discipline as a standard practice has implemented the single most effective AI cost control available in this generation of technology.

Section 3 · The Context Window and Working Memory

The whiteboard analogy

Every AI model has a context window. The context window is the total amount of text the model can consider at one time. It includes the system prompt, the conversation history, any documents you have attached, and any tool outputs the model has seen. It is, in effect, the working memory of the model during a single interaction.

> **The whiteboard**
> Imagine the model has a single whiteboard of fixed size. Everything written on the whiteboard is available to the model. Anything that does not fit on the whiteboard does not exist as far as the model is concerned. When the whiteboard fills up, the oldest content scrolls off the top edge and is forgotten. The size of the whiteboard is the context window. Different models have different sizes of whiteboard.

Why context window size matters

The context window of Claude in mid-2026 is two hundred thousand tokens, which corresponds to roughly five hundred pages of text. The context window of the most current ChatGPT models is similar in this range, though it varies by tier. This is enormous compared to where these models started just a few years ago, when four thousand tokens was the standard.

A larger context window allows the model to consider more material at once. A finance team can drop a full ten-K filing into a conversation and ask the model to analyze it. The model can hold the entire document in working memory simultaneously, alongside the question and any analytical framework you have provided. This capability, which would have been impossible three years ago, is the foundation of most of the workflows we will discuss in Parts 5 through 9.

The trade-offs of large context

There are trade-offs. Larger context costs more, because every token in the context is an input token that must be processed. A workflow that loads two hundred pages of context every time it runs is many times more expensive than a workflow that loads two pages. There is also a phenomenon called the lost-in-the-middle effect, where the model's attention to material in the middle of a very long context tends to weaken compared to material at the beginning and the end. The implication is that simply throwing more material into the context does not always produce better output. Sometimes a smaller, more focused context produces sharper analysis than a sprawling one.

This is a quiet but important design lesson. The largest possible context is not always the best choice. The right amount of context is the amount that is sufficient for the task, no more. Retrieval-augmented generation, which we will discuss in Section 9, exists precisely to give the model the right amount of context at the right moment, drawn from a much larger underlying corpus.

What happens when the context window fills up

When a conversation exceeds the context window, the oldest material is dropped. In a chat interface like ChatGPT or Claude, this happens silently. You may not notice that the model has forgotten the beginning of a long conversation. The model itself does not announce that it has forgotten anything. It simply continues generating, working only with what remains in its window.

For a CFO, this has two practical implications. First, in long analytical conversations, important constraints established early in the chat may become invisible to the model by the end. The CFO who notices the model contradicting earlier guidance is often observing context loss. The remedy is to re-state the critical constraints periodically, or to start a fresh conversation with the relevant context loaded at the top.

Second, for any production workflow, the context management is a design decision that must be made explicitly. How much of the prior conversation does the workflow retain? Which documents are loaded? In what order? How are they updated? These are architectural choices that affect both quality and cost. A well-designed workflow makes these choices deliberately. A poorly designed workflow drifts.

Context is not memory

It is worth pausing to emphasize a distinction that confuses many first-time users of these systems. The context window is not the same as memory. The context window is what the model can see in this exact moment. Memory, in the sense of remembering you across conversations, is a separate system that must be deliberately built and engaged.

In recent versions of ChatGPT and Claude, persistent memory features have been introduced that allow the system to retain certain facts across conversations. These features are useful, but they are constructed, not native. They work by saving information to a small store and re-injecting that information into the context window of future conversations. The model itself does not remember. Something around the model is helping it pretend to remember by reconstructing the context.

This distinction matters because, for any enterprise workflow, the question of what gets remembered, who controls what is remembered, and how the remembered material is governed is a design and policy question. It is not something the model handles on its own. The CFO who deploys AI in finance must decide what the architecture remembers and what it forgets, and must build that decision into the workflow explicitly.

> **The executive principle**
>
> Treat the context window as the model's mind in the present moment. Treat memory as a deliberate engineering choice that lives outside the model. Conflating the two leads to expensive surprises.

Section 4 · Prompts and the Discipline of Asking Well

The quality of the answer is determined by the quality of the question

Every interaction with a large language model begins with a prompt. The prompt is the text you send to the model. It can be a single sentence. It can be a multi-page document. It can be a structured set of instructions, examples, constraints, and reference material. The structure of the prompt determines the structure of the output.

The first lesson of working with these models is that the model is a very good interpreter of unclear instructions, which is both a strength and a trap. The model will produce something in response to almost any prompt, but the quality of what it produces is sharply sensitive to the quality of the prompt. A vague prompt produces a vague answer. A prompt that specifies role, context, task, format, and constraints produces a precise answer.

The five elements of a strong prompt

Across thousands of workflows in finance, a consistent pattern emerges. Strong prompts contain five elements, in roughly the following order:

1. Role

The role tells the model who it is acting as. "You are a senior FP&A; analyst preparing a board package for a Series B SaaS company." This single sentence shapes everything that follows. It establishes vocabulary, register, depth, and assumptions. The model is not actually an FP&A; analyst, but it has read enough FP&A; analysis to imitate the patterns of one when instructed.

2. Context

The context provides the situational background the model needs to interpret the task correctly. "The company has $22M ARR, grew 47% year over year, and missed Q3 by 8% due to a slipped enterprise deal. The board meeting is next Tuesday." Without context, the model produces generic output. With context, the model produces output that is grounded in the specific situation.

3. Task

The task is the specific instruction. "Draft the operating review section of the board package, focusing on the Q3 miss, the recovery plan, and the implications for the Q4 forecast." The task should be one sentence, and it should be specific enough that you could imagine evaluating whether the output meets the task.

4. Format

The format tells the model what shape the output should take. "Produce six paragraphs of executive prose, no bullet points, with the first paragraph framing the situation, the next three paragraphs analyzing the miss, and the final two paragraphs outlining the recovery." A specified format constrains the model in ways that consistently improve the output.

5. Constraints

Constraints are the rules the output must follow. "Do not use any forward-looking statements that have not been pre-approved. Avoid forecasting specific Q4 numbers. Reference the recovery plan but do not commit to it as guidance. Maintain the tone of the prior board commentary, samples of which are attached." Constraints are what differentiate executive-grade output from generic output. The model is excellent at honoring explicit constraints. The model is poor at inferring constraints that have not been stated.

> **The five-element discipline in one sentence**
>
> Role, context, task, format, constraints. Every serious finance prompt should contain all five. Most prompts in practice contain one or two. The gap between a prompt with one element and a prompt with five elements is the gap between disposable AI output and reliable AI output.

A side-by-side example

Consider the difference between three versions of the same request. The first version is what most executives type when they first encounter an AI system. The second is a moderate improvement. The third is what a CFO who has internalized the discipline of prompting would write.

Version 1: weak prompt

"Write a board commentary about Q3 results."

The model will produce a generic block of board commentary. It will reference unspecified financial results, use plausible-sounding metrics, and apply a tone that may or may not match the company's actual board culture. The output is essentially unusable except as a rough template.

Version 2: moderate prompt

"Write a board commentary about Q3 results for a Series B SaaS company that missed its number. Keep it professional and around 500 words."

The output improves materially. The model now has enough context to choose appropriate vocabulary and structure. But the output is still generic in important ways. The model is inventing the specifics of the miss, the recovery, and the forecast, because no specifics have been provided.

Version 3: strong prompt

"You are the CFO of a Series B B2B SaaS company preparing the Q3 operating review for the board. The company has $22M ARR, grew 47% year over year, and missed Q3 by 8% due to a slipped $1.2M enterprise deal that has now moved to Q4. NDR was 118%. New ARR was $1.8M against a plan of $2.4M. Cash runway is 22 months. The board chair has previously expressed concern about pipeline reliability. Draft the operating review in six paragraphs of executive prose, no bullet points. The first paragraph frames the quarter. The next three analyze the miss, explain the slipped deal, and address pipeline reliability honestly. The final two outline the recovery and set up Q4 without committing to specific numbers. Maintain a measured tone consistent with prior commentary, which I will attach. Avoid forward guidance and avoid defensive language. Acknowledge what went wrong and what we are doing about it."

The output from this prompt is substantively useful. It is not finished, because no AI output is finished, but it is a genuine starting point for serious work. The CFO who learns to prompt this way unlocks a different category of value from these systems than the CFO who treats them as a search engine.

System prompts versus user prompts

A small but important distinction. In the chat interface, the prompt you type is called a user prompt. Behind the scenes, there is also a system prompt, which is configured by the developer who built the chat interface and which sets the overall behavior of the assistant. The system prompt is invisible to most users.

For workflows that you or your team build directly using the API, you have control over both the system prompt and the user prompt. The system prompt is where you place the persistent rules: who the assistant is, how it should behave, what it must never do, what tone it must maintain. The user prompt is where the individual request goes. Separating these cleanly is a sign of a mature prompt architecture. Mixing them is a sign of a workflow that will be hard to maintain.

The discipline of iteration

Even with the best prompt design, the first output from a model is rarely the best output. Mature prompting includes deliberate iteration. You provide the prompt. You evaluate the output. You identify where it falls short. You add a refinement to the prompt that addresses the shortfall. You re-run. You evaluate again. After two or three iterations, the prompt is usually stable and the output is usually consistent.

This iteration is the actual work of prompt engineering. The term sounds technical but the practice is essentially editorial. You are negotiating with a very fluent but very literal apprentice until the apprentice produces output that meets your standard. The skill is patience and precision in articulating what you want.

> *Mature prompting is not a technical skill. It is the executive discipline of saying clearly what you want.*

Section 5 · The API and How Systems Talk to AI

From chat to programmatic access

Most finance executives encounter AI through a chat interface. You open ChatGPT or Claude in a browser, type a message, and a response appears. This is the consumer experience, and it is a genuinely useful starting point. But the chat interface is only one way to reach the underlying model. The other way, and the way that matters for enterprise workflows, is through an API.

API stands for application programming interface. The name is unfortunate because it sounds technical, but the concept is simple. An API is a doorway through which one piece of software can ask another piece of software to do something. When the finance team's ERP system needs to fetch the latest exchange rate, it calls an API at a foreign exchange data provider. When the CRM system needs to look up an account in the master data system, it calls an API. APIs are how modern software is connected. They are everywhere, and they have been for two decades.

> **The hotel front desk analogy**
>
> Think of a large hotel. You do not walk into the kitchen to get a meal, you call the front desk and the front desk communicates with the kitchen. You do not walk into the laundry room to get fresh towels, you call the front desk and the front desk handles it. The API is the front desk. You make a request. The request is routed to the right department inside the hotel. A response comes back to you. You do not need to know what the kitchen looks like or how the laundry is processed. You only need to know how to speak to the front desk.

What an API call looks like in practice

When a developer wants to use the Anthropic or OpenAI model from inside another piece of software, they send a structured request to the API. The request contains the model name, the system prompt, the conversation history, the user message, and a few configuration parameters such as the maximum length of the response and a parameter called temperature that controls how creative or how conservative the output should be. The API returns the model's response as structured data that the calling software can then use.

A finance leader does not need to write API calls. The finance leader needs to understand that this is the mechanism by which AI is integrated into every enterprise workflow. When the team proposes to build an AI-powered variance analysis tool, what they are proposing is a piece of software that gathers the relevant financial data, formats it into an API request, sends it to a model, and processes the response. The intelligence is in the model. The workflow logic, the data preparation, the output handling, and the error management are in the surrounding software.

Why this matters for cost and architecture

Three implications follow from understanding the API.

First, every API call costs money in tokens. The cost discipline we discussed in Section 2 applies at the level of each call. A workflow that makes a hundred thousand API calls per month at two thousand tokens each is a different beast from a workflow that makes ten thousand calls at five hundred tokens each. The CFO who looks only at the per-call price misses the volume multiplier.

Second, every API call has the latency of a network round trip plus the time the model takes to generate the response. For a modern model producing a short response, this is typically a few seconds. For a long response, it can be twenty or thirty seconds. A workflow that depends on real-time response to a human user must be designed differently from a workflow that runs in the background overnight. The architecture must account for latency from the beginning.

Third, every API call is a moment where something can go wrong. The network can fail. The provider can be slow. The model can return an unexpected error. Production-grade workflows must include error handling, retries, and fallback behavior. An amateur workflow assumes that every call succeeds. A professional workflow assumes that some calls will fail and is designed to handle the failure gracefully.

The relationship between the API and the chat interface

It is worth understanding that the chat interface you use as a consumer is itself a piece of software that calls the API on your behalf. When you type a message in ChatGPT, the browser sends your message to a server, the server calls the OpenAI API, the API returns a response, and the response is displayed in your browser. The chat interface is a wrapper around the API. Anything the chat interface does, a custom-built workflow can also do, with more control and more flexibility.

This is liberating once you internalize it. You are not limited to what the consumer chat interface offers. Anything you can imagine doing with AI, you can build, because the same model is available through the same API to anyone with a developer account. The only constraints are your imagination, your engineering resources, and your governance framework.

> *The chat interface is a wrapper around the API. Once you see that, the entire AI economy becomes legible.*

Section 6 · Getting Your API Keys from Anthropic and OpenAI

The keys are credentials

To use the API from either Anthropic or OpenAI, you need an API key. An API key is a long string of characters that uniquely identifies your account and authorizes your usage. Every API call you make is associated with the key, and every cent of usage is billed to the account that owns the key.

> **Treat the key like a password**
> An API key is a credential with full access to the account and its billing. If your key leaks, anyone who has it can consume your tokens and run up your bill until you notice and revoke it. Keys should never be shared in email, never be posted in a chat channel, never be committed to a code repository, never be embedded in a public-facing application. The standard practice is to store keys in a secrets manager and to reference them by name from the code that uses them.

Getting an Anthropic API key

The process for getting an Anthropic API key in mid-2026 is straightforward. You navigate to console.anthropic.com and create an account if you do not already have one. The account should be a workspace account, not a personal account, if the usage is for a company. Anthropic offers organizational workspaces that allow multiple users, role-based access, and consolidated billing. For any enterprise use, the organizational workspace is the correct choice.

Once inside the console, you navigate to the API keys section and generate a new key. The key will be shown once. You copy it immediately and store it in a secure location. If you lose the key, you cannot retrieve it. You must generate a new one and revoke the old one. This is intentional. The provider does not retain plaintext copies of your key for security reasons.

Anthropic offers usage tiers based on cumulative spending. New accounts start at a lower tier with lower rate limits. As spending accumulates and the account demonstrates legitimate usage, the tier increases automatically and rate limits become more generous. For enterprise customers, direct conversations with the Anthropic sales team can establish custom commercial agreements with reserved capacity, volume discounts, and enterprise support.

Getting an OpenAI API key

The process for OpenAI is similar in structure. You navigate to platform.openai.com and create an account or sign in. As with Anthropic, you should establish an organizational account for any company use. The organization-level account allows team members to be added with various permissions and provides consolidated billing and usage visibility.

Within the platform, you navigate to the API keys section and create a new secret key. As with Anthropic, the key is shown only once. You copy it and store it securely. OpenAI also offers usage tiers that increase as cumulative spending and account history grow. New accounts have lower rate limits that expand over time.

OpenAI also offers a separate enterprise product called ChatGPT Enterprise that includes various administrative features, data handling commitments, and a unified billing relationship. For larger organizations, the enterprise tier may be more appropriate than the developer API. The decision between consumer-style enterprise access and developer API access depends on whether the organization is primarily consuming pre-built AI features or primarily building custom workflows.

Setting up billing and budget controls

Both providers require a payment method to be added before API usage begins. Both providers offer the ability to set hard spending caps. This is one of the most important steps for any finance leader. Set the cap deliberately. Set it low enough that an accident does not produce a six-figure surprise. Set it high enough that legitimate usage is not interrupted. Revisit the cap monthly as actual usage patterns stabilize.

Both providers also offer usage alerts. Configure these as well. A typical setup includes an alert at fifty percent of the budget, another at seventy-five percent, and a hard stop at one hundred percent. This gives the finance team multiple opportunities to investigate before the budget is exhausted.

> **A CFO checklist for API account setup**
>
> Establish an organizational account, not a personal account. Add named team members with appropriate role-based access. Configure a hard spending cap. Configure usage alerts at multiple thresholds. Store the API key in a secrets manager, never in code. Establish a monthly review cadence for actual usage versus budgeted usage. Identify the named human owner of the account inside the finance organization.

What you can actually do once you have a key

Once you have an API key, you can call the model from any piece of software that can make an internet request. This includes custom-built workflows, spreadsheets with the right add-in, workflow automation platforms, and integration tools. The key is the universal credential. The number of places it can be used is essentially unlimited, which is precisely why it must be guarded carefully.

For a CFO, the practical reality is that the engineering team or the finance systems team will be the ones writing the actual code that uses the key. The CFO's role is to ensure that the key is governed properly, that usage is monitored, that costs are tracked against budget, and that the key never escapes the controlled environment in which it should live.

Section 7 · Connectors and How AI Plugs Into Your Systems

The plug and the outlet

In their basic form, AI models do not have access to your company's data. They have access only to what is placed inside the conversation. If you want the model to answer a question about your sales pipeline, the pipeline data has to get into the conversation somehow. The simplest way is to copy and paste it. This works for one-off analysis. It does not work for any workflow that needs to run regularly.

A connector is a mechanism that allows the AI system to draw data from a specific source without anyone copying and pasting. When you connect Google Drive to Claude, you are establishing a connector. The model can now read documents from your Google Drive on demand, with your authorization. When you connect Salesforce to ChatGPT, the model can now query opportunity records. The connector is the wiring between the model and the system that holds your data.

> **The power strip**
> Think of the AI model as an appliance that can do many things, but only if it is plugged into a source of electricity. The connector is the outlet. Different connectors plug into different sources. A Google Drive connector plugs into your documents. A Salesforce connector plugs into your CRM. A Gmail connector plugs into your email. The model itself is the same in every case. What changes is what the model is plugged into.

What connectors exist today

In mid-2026, the major AI providers offer connectors for the most common enterprise systems. Anthropic's Claude offers connectors for Google Drive, Gmail, Google Calendar, Slack, GitHub, Jira, Salesforce, HubSpot, and many others. OpenAI's ChatGPT offers a similar list. The connectors are typically enabled through a settings panel in the consumer chat interface, or through configuration in the developer API.

Each connector requires authorization. When you connect Google Drive to Claude, you go through an authorization flow in which Google asks you to confirm that Claude is allowed to access your Drive on your behalf. The authorization can be limited to specific folders, specific accounts, or specific permission levels. The authorization can also be revoked at any time. The control sits with you, not with the AI provider.

What connectors can and cannot do

A connector enables the model to read data from the connected system on demand. When you ask Claude to summarize a document from your Drive, Claude's software fetches the document, places it into the conversation context, and the model then processes it as part of the prompt. The connector handles the fetching. The model handles the analysis.

A connector can also enable the model to take actions in the connected system, if the connector is configured to allow this. A Salesforce connector can be configured to let the model create a new note on an opportunity, or update a field on a contact. A Slack connector can be configured to let the model send a message in a channel. These write-permission capabilities are where governance becomes critical. A model with write access to a production system is a model that can do real damage if its output is poorly controlled.

> **The read-only default**
>
> For any finance application, the default starting posture should be read-only connectors. The model can see, but cannot change. Write permissions should be added only when a specific workflow justifies them, with explicit governance around what the model is allowed to write, when it is allowed to write, and what human approval is required before the write takes effect.

Connector limitations

Connectors have limits that are not always obvious. A connector is typically built for a specific product, by a specific provider, in a specific configuration. A Salesforce connector built for the standard Salesforce data model may not work cleanly with a heavily customized Salesforce instance. A connector that supports the most common API endpoints may not support the less common ones. The connector is a piece of engineering, and like any piece of engineering, it has a scope.

When evaluating an AI workflow that depends on a connector, the right question is not whether the connector exists. The right question is whether the connector supports the specific data and the specific operations the workflow requires. This is the kind of question a CFO should learn to ask early in any AI initiative, before the engineering effort has committed to an architecture that the connector cannot ultimately support.

Section 8 · The Model Context Protocol

The next generation of connector architecture

Connectors as we have just described them are a reasonable first-generation solution to the problem of getting AI models connected to enterprise systems. Each provider builds their own connectors for the systems their customers most commonly use. This works, but it does not scale. There are tens of thousands of enterprise systems in the world. No AI provider can build connectors for all of them.

The Model Context Protocol, or MCP, is the architectural response to this problem. MCP was introduced by Anthropic in late 2024 and has since been adopted broadly across the industry. The protocol defines a standard way for any AI system to communicate with any other system. Once a system exposes itself through MCP, any AI model that supports the protocol can connect to it. The integration becomes universal rather than custom.

> **The universal adapter**
> Before MCP, each AI provider built custom connectors for each enterprise system. This is like every appliance manufacturer making its own custom plug shape, requiring every wall to have many different outlet types. MCP is the equivalent of an industry standard for plugs. Once a system speaks the protocol, any AI that speaks the same protocol can connect to it. The integration becomes plug-and-play in a way that the older connector model never achieved.

Why MCP matters strategically

For a CFO making technology decisions today, MCP matters for three reasons.

First, MCP reduces vendor lock-in. A workflow that connects to a finance system through MCP is not bound to a single AI provider. The underlying model can be swapped from Anthropic to OpenAI to a future provider without rebuilding the integration. This is a meaningful protection against vendor risk, which we will return to in Part 4 on governance.

Second, MCP makes the long tail of enterprise systems addressable. Specialized finance systems that no AI provider would have built a custom connector for can now be made accessible to AI through MCP. The CFO who is currently locked out of AI workflows because their core systems are too specialized may find that the door has just opened.

Third, MCP creates a marketplace of integrations. Once the protocol becomes a standard, third parties begin building MCP servers for systems they support. The ecosystem grows faster than any single AI provider could grow alone. This is the same dynamic that made the web grow faster than any single platform,

and the same dynamic that made USB displace dozens of proprietary cable standards.

What an MCP server actually is

In technical terms, an MCP server is a small piece of software that sits in front of an enterprise system and exposes that system's capabilities to AI models through the MCP protocol. The server might expose a set of tools the model can use, a set of resources the model can read, or a set of prompts the model can use as starting points for common tasks.

A CFO does not need to know how to build an MCP server. The CFO needs to know that MCP servers exist, that they are the modern architectural pattern for AI integration, and that any enterprise AI initiative in 2026 and beyond should be evaluated in part on whether it uses MCP or something equivalent. A system that requires custom connectors for every integration is a system that is solving a problem that has already been solved by a better architecture.

Connectors and MCP together

It is worth saying clearly that connectors and MCP are not in opposition. In practice, connectors are increasingly built using MCP under the hood. The user-facing experience may still be called a connector, but the underlying mechanism is increasingly the protocol. This convergence is good for the CFO, because it means the architectural foundation is becoming more uniform across providers, which makes future migration and vendor substitution easier.

When a vendor proposes an AI integration that uses MCP, the right CFO response is approval. When a vendor proposes a fully custom integration that does not use any standard protocol, the right CFO response is to ask what the substitution path looks like if the vendor relationship changes.

Section 9 · Retrieval-Augmented Generation

The most misunderstood concept in enterprise AI

Retrieval-augmented generation, abbreviated RAG, is the single most important architectural pattern in enterprise AI today. It is also the one most often misunderstood, misrepresented in sales pitches, and confused with adjacent concepts. A CFO who understands RAG can evaluate ninety percent of enterprise AI proposals without needing further technical guidance.

> **The research assistant analogy**
>
> Imagine you have a research assistant who is very well educated and reads quickly but does not have any specific knowledge of your company. When you ask the assistant a question about your company, they cannot answer from their general education. They need to consult your specific documents. So they walk into your filing room, pull the three or four documents most relevant to your question, read them, and then synthesize an answer using both their general expertise and your specific documents. The walking and pulling is retrieval. The reading and synthesizing is generation. Together, they are retrieval-augmented generation.

Why RAG exists

A large language model knows what it was trained on. Anthropic and OpenAI train their models on enormous quantities of public text. The models do not know your company's specific revenue recognition policy. They do not know which deals are in your pipeline. They do not know which vendors you use. They cannot answer questions about your specific business unless that specific business information is brought into the conversation.

You could simply paste your entire revenue recognition policy into every prompt that asks about revenue recognition. This is the brute force approach. It works, but it is expensive in tokens, slow, and unwieldy. RAG is the elegant alternative. Instead of pasting everything, you maintain a searchable collection of your company's relevant documents and let the system retrieve the most relevant pieces on demand.

How RAG works in practice

A RAG system has three components. The first is the corpus, which is the collection of documents the system can draw from. For a finance application, the corpus might include accounting policies, prior period financial statements, board commentary archives, audit memos, contract templates, vendor master data, and so on. The corpus is your institutional knowledge in searchable form.

The second component is the retrieval system. When a question is asked, the retrieval system identifies the documents or sections of documents most relevant to the question. The most common technique uses what are called embeddings, which are mathematical representations of the meaning of text. The retrieval system converts the question into an embedding, compares it to the embeddings of all the documents in the corpus, and returns the closest matches. This is, in effect, semantic search that operates on meaning rather than on keyword matching.

The third component is the generation step. The retrieved documents are placed into the context window of the model, along with the original question. The model then generates a response that draws on both its general knowledge and the specific retrieved material. The output is grounded in your documents, not in the model's training.

Why RAG is the right architecture for most finance use cases

RAG solves the problem that confronts every enterprise AI project. The model needs your data to be useful. Your data is too large to fit in every prompt and too dynamic to be trained into the model. RAG provides the bridge. Your data lives in your systems, gets retrieved on demand, gets brought into the context only when needed, and gets refreshed whenever the underlying documents change.

Almost every serious finance AI application uses some form of RAG. The board commentary generator retrieves prior commentary and prior financial statements. The variance analysis tool retrieves the relevant period's data. The policy compliance checker retrieves the relevant policy documents. The pattern is so common that it has become invisible. The CFO who recognizes RAG when they see it can evaluate proposals with much more clarity.

What can go wrong with RAG

RAG is not magic. Several things can go wrong. The retrieval can return the wrong documents, in which case the model generates a confident answer based on irrelevant material. The retrieval can miss relevant documents, in which case the model generates an answer that omits important context. The corpus itself can be out of date, in which case the model generates an answer that reflects yesterday's policy rather than today's. The corpus can contain contradictory documents, in which case the model has to choose which to trust, and the choice may not be the right one.

A well-designed RAG system addresses these failure modes explicitly. The retrieval system is tuned and tested. The corpus is curated, with clear ownership and a refresh schedule. The output is cited, so the human reader can see which documents the model drew from. The system is monitored for cases where the retrieval clearly failed. These are not features that come for free. They are engineering work that someone has to do.

What to ask about any RAG proposal

Who owns the corpus and keeps it current? How is the retrieval quality measured? Does the output cite the documents it drew from? What happens when the retrieval fails? Can the user see and verify which documents were retrieved? These are the questions that separate production-grade RAG from demo-grade RAG.

Section 10 · Fine-Tuning, RAG, and Prompting: The Architectural Choice

Three ways to make a model do what you want

When you want a model to behave a certain way for your specific use case, you have three available architectures. The choice among them is one of the most consequential technical decisions in any enterprise AI initiative. A CFO who understands the choice can evaluate engineering proposals with confidence.

Prompting

You give the model instructions and information in the prompt itself. The model is unchanged. The behavior is configured entirely through what you put in the conversation. This is the most flexible approach, the cheapest to implement, and the fastest to change. For most finance use cases, prompting alone gets you most of the way there.

RAG

You give the model your specific documents at runtime through a retrieval system, as we have just discussed. The model is still unchanged, but its context is enriched with your institutional knowledge. RAG is the right architecture when the model needs to answer questions grounded in a body of company-specific material that is too large to fit in a single prompt.

Fine-tuning

You actually modify the model itself by training it further on examples of the behavior you want. The fine-tuned model is a new model, derived from the original but adjusted for your specific patterns of output. Fine-tuning is the most powerful of the three approaches, but it is also the most expensive, the slowest to implement, and the hardest to maintain.

When to use each

The honest answer for most finance use cases is that prompting plus RAG is sufficient. Prompting alone handles the cases where the model's general training is adequate and the only need is to direct its output. RAG handles the cases where the model needs grounding in specific institutional knowledge. The combination of the two covers ninety percent of practical finance workflows.

Fine-tuning becomes worth considering only in narrow circumstances. When you have a very specific output format that the model struggles to produce reliably through prompting alone. When you have a large body of examples that demonstrate the exact pattern you want and prompting cannot capture it. When the cost of running prompting plus RAG at scale has become so large that the upfront cost of fine-tuning is justified

by the ongoing savings. These are real cases, but they are specialized cases. Most finance teams should not be fine-tuning in their first two years of AI deployment.

> **The order of preference**
> For any new use case, the default should be prompting first. If prompting alone is insufficient, add RAG. If prompting plus RAG is still insufficient, only then consider fine-tuning. Skipping the prompting and RAG steps and going straight to fine-tuning is a common error and an expensive one. It is the AI equivalent of buying a custom-built machine when an off-the-shelf machine plus an attachment would have done the job.

The cost and time profile of each

Prompting can be iterated in minutes. The cost is whatever the API call costs. The time from idea to production is hours or days.

RAG can be iterated in weeks. The cost includes the embedding and storage of the corpus, plus the API calls. The time from idea to production is several weeks for a well-scoped use case, several months for an enterprise-grade deployment.

Fine-tuning is iterated in months. The cost includes the training run, which can range from hundreds of dollars to tens of thousands of dollars depending on the size of the model and the volume of training data. The fine-tuned model also has higher per-call costs at inference time. The time from idea to production is several months at minimum, and the resulting model needs to be maintained as the underlying base model evolves.

The CFO who hears a proposal to fine-tune a model in the first phase of an AI initiative should ask hard questions about why prompting and RAG are not being tried first.

Section 11 · Workflows and Agents

The two patterns of AI deployment

Once you have a model, an API, connectors or MCP, and a grounding mechanism such as RAG, you can build AI into your business operations in two distinct patterns. These two patterns differ in their governance posture, their predictability, their risk profile, and their implementation complexity. The CFO needs to understand both and to know which one is appropriate for which situation.

Workflows

A workflow is a defined sequence of steps. The sequence is designed in advance. Each step has a defined input, a defined operation, and a defined output. The AI model is invoked at specific points in the sequence to perform specific operations: analyze this data, summarize this document, classify this transaction, draft this commentary. The flow of control is managed by software, not by the model. The model is a component within the workflow, not the orchestrator of it.

Agents

An agent is an AI system that orchestrates its own sequence of steps. The agent is given a goal, a set of tools it can use, and a degree of autonomy. The agent decides what to do, when to use which tool, how to interpret intermediate results, and when the goal has been achieved. The flow of control is managed by the model itself. The agent is the orchestrator.

The trade-offs

Workflows are predictable, auditable, and easy to govern. The designer of the workflow knows exactly what will happen in each step. The audit trail is clear. The failure modes are limited to the failure modes of each component. If a workflow misbehaves, the misbehavior is localized to a specific step, and the step can be inspected and corrected.

Workflows are also less flexible. A workflow built for one task does not handle adjacent tasks gracefully. Adding new capabilities requires changing the workflow design, which requires engineering work and testing. For tasks that are well-defined and stable, this is acceptable. For tasks that vary considerably from instance to instance, workflows become rigid.

Agents are flexible. The same agent, given different goals, can tackle different tasks. The agent reasons about the situation and selects the appropriate tools. This adaptability is what makes agents exciting. It is also what makes them dangerous.

Agents are harder to govern. The audit trail is more complex, because the sequence of steps is not predetermined. The failure modes are broader, because the agent can compose tools in ways the designer did

not anticipate. When an agent misbehaves, the misbehavior may not be localized. The agent may have used a tool incorrectly, or used the wrong tool, or interpreted a result incorrectly, or chained several steps in a way that produced a bad outcome through no single bad action.

> **The principle for finance applications**
> For finance applications in 2026, workflows are the default. Agents are exceptional. The reason is that finance requires auditability, predictability, and traceability at a level that workflows naturally support and that agents require considerable additional engineering to provide. As agent technology matures and as governance tools catch up, this balance will shift. But in the present moment, the CFO who deploys workflows by default and agents only with specific justification is making the right architectural choice.

Hybrid patterns

In practice, many real systems combine workflows and agents. A workflow may delegate a single complex step to a small agent. An agent may invoke a deterministic workflow as one of its tools. The boundary is not always sharp. What matters is that the CFO understands which parts of the system are deterministic and which parts involve model judgment, and governs them accordingly.

Every step in a finance system that involves model judgment is a step that needs to be designed with the assumption that the judgment may occasionally be wrong. The question is not whether the model will sometimes err. The question is whether the system handles the error gracefully when it occurs.

The vocabulary problem

The vocabulary around AI workflows and agents is in active evolution and is often confused. Vendors use the word agent to mean many different things, from a chat interface with a few tools attached, to a complex multi-step autonomous system, to almost anything in between. The CFO should not be impressed or alarmed by the word itself. The right question is what the system actually does, what tools it has, what decisions it makes autonomously, and what the audit trail looks like.

When a vendor proposes an agent for your finance operations, ask them to draw the diagram. Ask them to show every tool the agent can call. Ask them to show what happens when the agent makes a wrong decision. Ask them to show the audit log of a representative session. The answers to these questions will tell you more about the agent than any marketing language will.

Section 12 · Governance, Security, and Risk

Why this section is brief

Part 4 of this masterclass is devoted entirely to governance. This section in Part 1 is a brief introduction to the territory, intended to ensure that the CFO finishes the foundation with the right instincts about risk. The full treatment, the methodology, and the operational framework come in Part 4.

The governance perimeter

Every AI deployment in a finance organization sits inside a governance perimeter. The perimeter is defined by data, by models, by outputs, by people, and by regulations. Each of these dimensions deserves attention.

Data

Which data is the model allowed to see? Which is it not? Where does the data go when it enters the model's context? Does the provider retain it? For how long? Who at the provider can see it? What jurisdictions does it cross? Each of these questions has answers that vary by provider, by product tier, by contract. The CFO should know the answers for any AI system their organization uses.

Models

Which model is being used for which workflow? What version? How is the version controlled? What happens when the provider releases a new version? How are the workflows tested against the new version before cutover? Model versioning is a category of change management that finance organizations have rarely had to consider, and that is becoming a regular discipline.

Outputs

What is the model allowed to produce? What review process applies before the output is used? Is the output ever sent directly to an external party without human review? Is it ever used to make a financial entry without human approval? The design of the output gating is among the most consequential decisions in any finance AI deployment.

People

Who is the named human owner of each AI workflow? Who is authorized to invoke it? Who approves its outputs? Who is accountable when it produces an error? Without named accountability, AI workflows drift into ambiguity, and ambiguity in a finance organization is itself a control weakness.

Regulations

What regulations apply? SOX for public companies in the United States. GDPR for personal data of European individuals. The EU AI Act for systems deployed in or affecting the European Union. Sectoral regulations in financial services, healthcare, and other industries. The regulatory landscape is evolving rapidly, and the CFO must maintain a current understanding of what applies.

The instincts you should carry forward

Three instincts are worth establishing now, before we begin using the technology in earnest.

First, AI in finance is not magic. It is a tool. Tools require discipline, training, and governance. The discipline does not reduce the value of the tool. The discipline is what makes the value reliable.

Second, the most expensive error in finance AI is the one that is invisible. A workflow that quietly produces slightly wrong output for months is more damaging than a workflow that fails noisily on day one. Governance is, in significant part, the discipline of making errors visible.

Third, the CFO is the right person to govern AI in finance. Not the chief information officer, not the chief technology officer, not the chief data officer, although each of them has a role. The CFO is the executive whose function is most transformed by this technology, whose accountability is most directly engaged by its outputs, and whose authority is most naturally aligned with the discipline required. This is not a function the CFO should delegate. It is a function the CFO should own.

> *AI in finance is not magic. It is a tool. Tools require discipline, training, and governance. The discipline does not reduce the value. The discipline is what makes the value reliable.*

The foundation is now complete. You have a working understanding of what a large language model is, how it is priced, how it is accessed, how it is integrated with enterprise systems, how it is grounded in your specific data, what architectural patterns are available, and where the governance perimeter sits. This is enough to begin the discovery process that John Campbell undertakes in Part 2.

Before you proceed, take the assessment that follows. It is twenty questions designed to test whether the foundation is actually solid. The answers, with brief explanations, follow the assessment. Use them honestly.

Appendix A · Glossary

A consolidated reference of the terms introduced in this part. You may return to this glossary at any point throughout the masterclass.

Agent
An AI system that orchestrates its own sequence of steps to achieve a goal, deciding which tools to use and in what order. Contrast with workflow.

API (Application Programming Interface)
The doorway through which one piece of software requests services from another. In AI, the API is how a custom application invokes a model.

API key
A long string of characters that uniquely identifies an account and authorizes usage of the API. Must be guarded like a password.

Connector
A mechanism that allows an AI system to draw data from a specific enterprise system, such as Google Drive or Salesforce, with authorization.

Context window
The total amount of text a model can consider at one time, including the system prompt, conversation history, and any attached material. Measured in tokens.

Embedding
A mathematical representation of the meaning of a piece of text, used by retrieval systems to find semantically similar content.

Fine-tuning
The process of further training a base model on examples of the behavior you want, producing a new model derived from the original.

Generation
The process by which a model produces output text, one token at a time, by predicting the next most likely token given everything that has come before.

Grounding
The practice of providing the model with the specific source material it needs to answer accurately, rather than relying on its general training.

Hallucination
The production of confident but incorrect output by a model. Arises from the predictive nature of the model, which generates plausible-sounding text without verifying its accuracy.

Input tokens
The tokens sent to the model, including the system prompt, conversation history, and user message. Typically priced lower than output tokens.

Large language model (LLM)
A statistical model trained on large quantities of text that predicts the next likely token given a sequence of preceding tokens.

Latency
The time between sending a request and receiving a response. For language models, latency depends on the length of the output and the model's generation speed.

Model Context Protocol (MCP)
A standardized way for AI models to communicate with enterprise systems and tools, introduced by Anthropic in late 2024 and adopted broadly.

MCP server
A small piece of software that exposes an enterprise system's capabilities to AI models through the Model Context Protocol.

Output tokens
The tokens generated by the model in response to a prompt. Typically priced higher than input tokens because generation is computationally more expensive.

Prompt
The text sent to the model that instructs it what to do. May include role, context, task, format, and constraints.

RAG (Retrieval-Augmented Generation)
An architecture in which the model retrieves relevant documents from a corpus at runtime and uses them as context for generating its response.

Retrieval
The first step of a RAG system, in which the documents most relevant to a question are identified and fetched from the corpus.

System prompt
The persistent set of instructions that configures the model's behavior for an entire session, separate from the user's individual messages.

Temperature
A configuration parameter that controls how creative or how conservative the model's output is. Lower temperature produces more predictable output; higher temperature produces more varied output.

Token
The unit of text the model processes, typically corresponding to three-quarters of an English word or about four characters. The currency of the AI economy.

Tokenizer
The software that breaks text into tokens for the model to process.

Workflow
A defined sequence of steps that the AI system executes. The flow of control is managed by software, with the model invoked at specific points. Contrast with agent.

Appendix B · Assessment

Twenty questions to test the foundation. Twelve multiple choice, five short answer, and three scenario-based. The answer key with explanations follows in Appendix C.

Answer the questions on your own first. Then check the answer key. If you answer fifteen or more correctly, the foundation is solid and you are ready to proceed to Part 2. If you answer fewer than fifteen correctly, return to the relevant sections before moving on. The cost of a weak foundation is paid throughout the remaining nine parts of the masterclass.

Part I: Multiple Choice (Questions 1–12)

1. A large language model produces output by:

(a) Looking up answers from a structured database it was given access to during training.
(b) Predicting the next most likely token given everything that has come before.
(c) Searching the live web for current information.
(d) Consulting a knowledge graph built into the model architecture.

2. Which of the following is the most accurate description of a token?

(a) A unit of currency used to access AI providers.
(b) A small chunk of text, typically about three-quarters of an English word, that is the basic unit a model processes.
(c) A complete sentence, used as the smallest meaningful unit in language models.
(d) A unique identifier assigned to each conversation in the API.

3. The context window of an AI model refers to:

(a) The web browser used to access the AI chat interface.
(b) The total amount of text the model can consider at one time, including system prompt, conversation history, and attachments.
(c) The hours of the day during which the API is available.
(d) The geographic regions where the model is deployed.

4. Output tokens are typically priced higher than input tokens because:

(a) Output tokens are larger units of text than input tokens.
(b) Generating new text is computationally more expensive than reading existing text.
(c) AI providers prefer customers who consume less output.
(d) Output tokens consume more storage space.

5. The five elements of a strong prompt, in the order presented in this part, are:

(a) Role, context, task, format, constraints.

(b) Question, answer, evidence, conclusion, recommendation.

(c) Subject, verb, object, modifier, qualifier.

(d) Greeting, request, justification, deadline, signoff.

6. An API key should be treated like:

(a) A username, freely shared within the team.

(b) A password, guarded carefully and never embedded in code or shared in email.

(c) A serial number, displayed on hardware for support purposes.

(d) A version identifier, included in documentation.

7. A connector in the context of AI systems is best understood as:

(a) A power cable that physically links the AI server to the data center.

(b) A mechanism that allows an AI system to draw data from a specific enterprise system with authorization.

(c) A type of subscription tier offered by AI providers.

(d) A device used to broadcast AI responses to multiple users.

8. The Model Context Protocol, or MCP, is best described as:

(a) A new model architecture replacing transformers.

(b) A standardized way for AI models to communicate with enterprise systems and tools.

(c) A regulatory framework governing AI deployment.

(d) A proprietary technology owned exclusively by OpenAI.

9. Retrieval-augmented generation (RAG) is an architecture in which:

(a) The model is retrained on the company's data each night.

(b) Relevant documents from a corpus are retrieved at runtime and used as context for the model's response.

(c) Multiple AI models vote on the best answer.

(d) The model is augmented with additional parameters during fine-tuning.

10. Among prompting, RAG, and fine-tuning, the appropriate order of preference for most new finance use cases is:

(a) Fine-tuning first, then RAG, then prompting.

(b) Prompting first, then RAG if needed, then fine-tuning only with specific justification.

(c) RAG first, then fine-tuning, then prompting.

(d) All three should be deployed simultaneously for best results.

11. The primary difference between a workflow and an agent is:

(a) Workflows are faster than agents.

(b) A workflow follows a predefined sequence of steps; an agent decides its own sequence to achieve a goal.

(c) Workflows are more expensive than agents.

(d) Agents are only available to enterprise customers.

12. For most finance applications in 2026, the default architectural pattern should be:

(a) Agents, because they are more flexible.

(b) Workflows, because they are more predictable and easier to govern.

(c) Direct chat interfaces only, with no automation.

(d) Fine-tuned models, because they are more accurate.

Part II: Short Answer (Questions 13–17)

13. In two or three sentences, explain why hallucination is a consequence of how large language models generate text, rather than a defect that can be eliminated through better engineering.

14. A vendor offers an AI-powered document analysis service at $0.20 per document. Your finance team will process roughly 50,000 documents per month. In two or three sentences, explain what questions you would ask before accepting the quoted rate as the expected monthly cost.

15. Distinguish between the context window of a model and the persistent memory features that some AI products offer. Explain why this distinction matters for enterprise workflow design.

16. In two or three sentences, explain when fine-tuning is genuinely appropriate for a finance use case, and why most teams should not fine-tune in their first two years of AI deployment.

17. A vendor proposes an AI integration using a fully custom connector to your ERP system, not built on the Model Context Protocol. In two or three sentences, explain what concerns you would raise and what alternative you would prefer.

Part III: Scenario-Based (Questions 18–20)

18. Scenario: Your CRO has proposed an AI agent that will autonomously update Salesforce records based on sales call transcripts. The agent will have write access to opportunities, contacts, and accounts. In one paragraph of executive prose, describe the principal governance concerns you would raise as CFO and the conditions under which you would support or oppose the proposal.

19. Scenario: Your FP&A; team has built a board commentary generator that uses RAG over a corpus of prior board materials and current period financials. In its first three months of use, two of the generated commentaries contained subtle factual errors that the analyst on duty did not catch before the package was distributed to the board. In one paragraph, explain what architectural or process changes you would require before allowing the system to continue in production.

20. Scenario: A finance leader at a peer company tells you they have eliminated their FP&A; team by deploying autonomous AI agents that produce the monthly close, the variance analysis, the board pack, and the cash flow forecast without human involvement. They are saving substantial cost and have not experienced any issues in their first quarter. In one paragraph of executive prose, describe how you would respond to this claim and what posture you would take toward AI agents in your own finance function.

Appendix C · Answer Key with Explanations

Check your answers honestly. The brief explanations are intended to reinforce the concept, not merely to confirm the correct letter. If you got an answer wrong, return to the cited section and re-read it before moving on.

Multiple Choice Answers

Question 1: (b)

A large language model generates text by predicting the next most likely token. It does not look anything up in a database and does not search the live web in its base form. This predictive nature is the source of both the model's fluency and its hallucinations. See Section 1.

Question 2: (b)

A token is a small chunk of text, typically about three-quarters of an English word. Tokens are the unit by which AI usage is measured and priced. See Section 2.

Question 3: (b)

The context window is the model's working memory, encompassing everything the model can see at once. The model has no awareness of anything outside this window. See Section 3.

Question 4: (b)

Generation is computationally more expensive than reading, because each output token requires the model to compute probabilities across its entire vocabulary. This is why output tokens are priced three to five times higher than input tokens. See Section 2.

Question 5: (a)

Role, context, task, format, constraints. This is the discipline that separates executive-grade prompting from amateur prompting. See Section 4.

Question 6: (b)

An API key is a credential with full access to the account and its billing. Treating it like a password is the only appropriate posture. See Section 6.

Question 7: (b)

A connector is the wiring between an AI system and a specific enterprise system, enabling the AI to read and sometimes write data from that system with authorization. See Section 7.

Question 8: (b)

MCP is an open standard for AI-system communication, introduced by Anthropic and adopted broadly. It reduces vendor lock-in and makes integrations more universal. See Section 8.

Question 9: (b)

RAG combines retrieval of relevant documents with generation grounded in those documents. The model itself is unchanged; only the context provided to it is enriched. See Section 9.

Question 10: (b)

The order of preference is prompting, then RAG, then fine-tuning. Skipping straight to fine-tuning is a common and expensive error. See Section 10.

Question 11: (b)

A workflow follows a predefined sequence designed by humans. An agent decides its own sequence to pursue a goal. The governance implications of this difference are substantial. See Section 11.

Question 12: (b)

Workflows are the default for finance applications because they are predictable, auditable, and easier to govern. Agents are exceptional and require specific justification. See Section 11.

Short Answer Explanations

13. Hallucination as a consequence, not a defect

Hallucination arises because the model generates text by predicting plausible continuations, not by retrieving verified facts. When the model's training is well-aligned with the question, the prediction is accurate. When it is not, the model still generates a fluent, confident answer because it has no mechanism for representing uncertainty. Engineering can reduce hallucination through grounding, RAG, and verification steps, but it cannot eliminate the underlying mechanism that makes it possible. The CFO who understands this designs workflows that verify rather than trust.

14. The $0.20 per document quote

The right questions concern token consumption. What is the average document size in tokens? What is the system prompt overhead per call? What output length does the analysis produce? At fifty thousand documents per month with an average of, say, three thousand input tokens and five hundred output tokens each, the actual token consumption may make the per-document cost higher than quoted. The CFO should also ask whether the $0.20 includes the underlying API costs or whether those are passed through separately. The gap between a quoted price and the all-in monthly bill is where most AI budget surprises live.

15. Context window versus persistent memory

The context window is the model's working memory in the present moment. It exists only during a single interaction and includes whatever is loaded into the conversation. Persistent memory features, by contrast, are constructed around the model: information is saved to an external store and re-injected into the context window of future conversations. The distinction matters because, for an enterprise workflow, the question of what gets remembered is a deliberate design choice, not a property of the model. The architecture team must decide what to retain, how to govern it, and how to retrieve it.

16. When fine-tuning is appropriate

Fine-tuning is appropriate when prompting plus RAG have been tried and found insufficient, when the team has a large corpus of high-quality examples of the desired output, and when the cost of running prompting plus RAG at scale justifies the upfront investment in training. Most teams should not fine-tune in their first two years because they have not yet exhausted what prompting and RAG can do, because they lack the example corpus needed to fine-tune well, and because fine-tuned models require ongoing maintenance as base models evolve. Skipping straight to fine-tuning is almost always premature optimization.

17. The custom connector concern

The principal concern is vendor lock-in and the risk that the integration cannot be substituted if the vendor relationship changes, the vendor's product evolves, or the underlying ERP changes. A custom connector also means that the integration is bespoke engineering that must be maintained by someone, with no broader ecosystem of compatible tools. The preferred alternative is an MCP-based integration, which is portable across AI providers and maintained as part of a growing standard. If the vendor insists on a custom approach, the CFO should require a documented substitution path and contractual protections against future migration cost.

Scenario Discussions

18. The autonomous Salesforce agent

The principal concerns are at least four. First, write access to a system of record is the highest-risk category of AI integration, because errors are not contained within the AI system but propagate into the operational data. Second, an agent that decides its own sequence of actions is harder to audit than a workflow with predefined steps; every update should leave a clear trace of why it was made. Third, sales call transcripts are themselves noisy data, and an agent that updates records based on transcripts will sometimes update them incorrectly; the question is what happens then. Fourth, the CRO's motivation is reasonable but the governance burden falls on the CFO's office. The conditions for support would include a workflow-based design rather than an autonomous agent for any high-risk updates, a human approval gate for any write to opportunity amount, stage, or close date, an immutable audit log of every update, regular reconciliation against the source transcripts, and a defined process for reverting incorrect updates. Without these conditions, the proposal should not proceed in its current form.

19. The board commentary errors

The two errors are the warning, not the problem. The problem is that the architecture allowed errors to reach the board without being caught. Several changes are required. First, the system must cite the source documents for every claim in the generated commentary, so that a reviewer can verify the claim against the source rather than against memory. Second, the review process must include a structured verification step rather than a casual read-through; the analyst should be required to confirm each cited claim. Third, the corpus quality and refresh process must be examined, because retrieval errors often trace back to outdated or contradictory documents in the corpus. Fourth, the system should flag claims that have low retrieval confidence rather than presenting all claims with equal apparent authority. Until these changes are in place, the system should not be used to produce final board materials. A pilot mode in which the system produces a draft that the analyst rewrites independently is acceptable; a production mode in which the system's output is distributed with light review is not.

20. The peer who eliminated FP&A;

The right response is professional skepticism, not admiration. Several considerations apply. First, one quarter is not enough data to establish that the system works; the errors that escape autonomous systems often compound silently before becoming visible. Second, the cost savings depend on what value the eliminated humans were producing beyond the visible deliverables, which often includes judgment, institutional knowledge, and the catching of errors that would have caused larger problems downstream. Third, the audit, regulatory, and governance posture of a finance function that has eliminated human judgment from its core processes is fragile, and the consequences of a failure are asymmetric: small savings against the possibility of a material misstatement or a control failure. The right posture in your own finance function is to deploy AI aggressively to augment human judgment, to use workflows rather than autonomous agents for any process touching the books, and to retain the human expertise needed to catch errors and to govern the system itself. The peer may be right, but they are running an experiment whose outcome is not yet known, and finance is not a function in which one should be eager to be the experimental subject.

End of Part 1

AI Foundations for the Finance Executive

The foundation is now in place. In Part 2, you will meet John Campbell, the new CFO of Helix Cloud Systems, on his first morning in the role. Over the course of his first thirty days, he will conduct a series of structured conversations across the executive team. You will watch him practice the discipline of executive listening, the discipline of applying what you have just learned about AI, and the discipline of identifying the workflows that will eventually become the five use cases at the heart of this masterclass.

Carry the foundation forward. Every conversation in Part 2 will draw on concepts you have just acquired. Every note John Campbell writes in Part 3 will reference the architectural patterns introduced here. The masterclass is constructed cumulatively. The investment you have made in this part will compound across the nine that follow.