Power BI Copilot
A Practical Guide to What Copilot Sees, Understands, and Cannot Know

Andrea Benedetti

Senior Solution Engineer, Data & AI, Microsoft

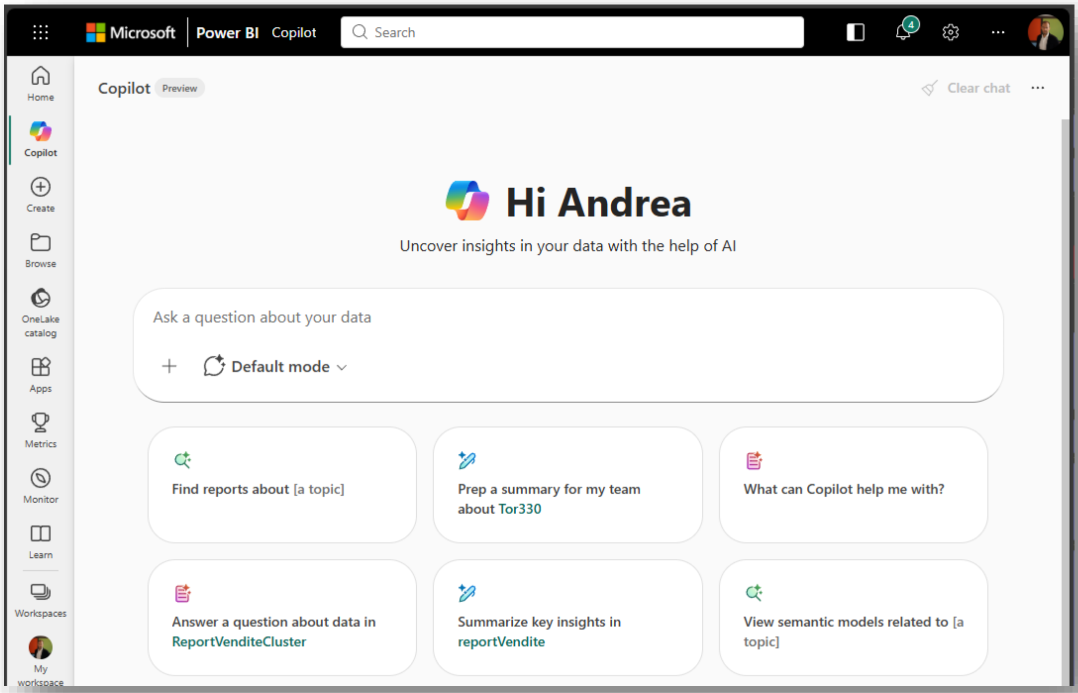
https://www.linkedin.com/in/abenedetti/

Introduction

This document aims to provide a clear, technical, and unfiltered overview of using Copilot in Power BI, helping Data & Analytics professionals, architects, and decision makers understand its real capabilities, structural limitations, and the necessary conditions for it to deliver tangible value in an enterprise context.

Copilot fundamentally changes how users interact with data. It lowers the barrier to exploration and accelerates ad hoc analysis, prototyping, and contextual queries.

However, its value is not automatic. Enabling Copilot does not equate to adopting it successfully, and Copilot does not "work on its own." Like any advanced capability, it requires the right context to deliver its full potential.


Microsoft
Power BI
Copilot
Search
Home
Copilot
Create
Browse
OneLake catalog
Apps
Metrics
Monitor
Learn
Workspaces
My workspace
Copilot Preview
Clear chat
Hi Andrea
Uncover insights in your data with the help of AI
Ask a question about your data
Default mode
Find reports about [a topic]
Prep a summary for my team about Tor330
What can Copilot help me with?
Answer a question about data in ReportVenditeCluster
Summarize key insights in reportVendite
View semantic models related to [a topic]

Copilot assumes the presence of solid semantic models, explicit business definitions, widespread skills, validation practices, user training, and a clear governance framework. Without these elements, the primary risk is operational rather than technological: unstable responses, ambiguous interpretations, and a perception of reliability that does not always align with the actual data. In this sense, Copilot is not ‘free’: its cost is not only computational, but also organizational, cultural, and architectural.

There is also a substantial difference compared to the past. Before the introduction of Copilot, semantic ambiguities and interpretation errors were mainly human and often silent, tied to the individual experience of those who built or consumed reports. With Copilot, these ambiguities become visible. Copilot does not introduce new uncertainty; it amplifies the semantics of the model it operates on: when the semantics are clear and well-coded, it amplifies clarity and consistency; when they are ambiguous or incomplete, it amplifies that same ambiguity. This effect can be uncomfortable, but it is precisely what makes Copilot valuable: it exposes limitations and inconsistencies that previously remained hidden, turning Copilot into a powerful stress-testing tool for the semantic model.

This document was created with the goal of making these dynamics explicit and providing practical guidance on how to design, use, and govern Copilot in Power BI consciously, to allow it to deliver real, reliable, and sustainable value over time.

1. What Is Copilot in Power BI (and What It Is NOT)

1.1 Correct Definition

Copilot in Power BI is not a single feature, but a set of integrated experiences (“Copilot experiences”) within Power BI/Fabric that leverage generative AI to:

- Consume content (summaries, guided Q&A, ad hoc visuals) from reports and semantic models;
- Create and edit content (report pages, narratives, DAX, measure descriptions);
- Guide users to find content (standalone/app-scoped) and interact with data using natural language.

In terms of “form,” it’s helpful to think of three main containers today:

- Copilot pane (report agent): the side panel inside a report
- Standalone Copilot (Power BI agent): full-screen experience that finds/analyzes any item the user has access to
- Copilot in apps (app agent): full-screen experience “scoped” to the content of a Power BI app

1.2 What It Can Do Today (Practically)

From a Data & Analytics team’s perspective, Copilot in Power BI should not be seen as a set of isolated functions, but as a coherent group of capabilities working across different layers of analytical experience.

The value is concentrated in three operational areas:

- Chat with data (for business users and advanced users)
    - “Answer questions about the data”
    - Access to transparency/validation via “How Copilot arrived at this”
    - Ability to add generated visuals to the page
- Support for report development
    - Create and edit report pages via prompts (add/change/delete visuals, undo/redo)
    - Generate narrative summaries
- Support for model/semantic layer development
    - Write/explain DAX queries (DAX query view) and create measure descriptions
    - Prepare the model for AI using AI data schema, AI instructions, verified answers, and “Approved for Copilot”

1.3 What It Does NOT Do (and why)

Copilot is not an “automatic analyst” that can generate advanced insights on demand.

For example, in the “Ask Copilot for data from your model” capability, it’s clear that it does not (currently) respond to requests involving anomaly detection, forecasting, or “key influencers”; these are examples of unsupported questions (“Why…?”, “How many … next year?”).

Copilot is not a replacement for people who model and build reports.

The documentation states this unambiguously: Copilot aims to increase efficiency and quality, not to replace those who build semantic models and reports; the generated results do not substitute the context and problem understanding that an analyst/BI developer has.

Copilot does not “understand the data” beyond what the semantic model allows it to understand.

If a question is “related to data in the semantic model,” Copilot uses the semantic model.

2. Conceptual Architecture: What Happens “Behind the Scenes”

2.1 The Semantic Model Is the Center of Everything

In Power BI, Copilot “reasons” mainly through what the semantic model exposes:

- Schema (tables, columns, measures, hierarchies, properties)
- Relationships and directionality
- DAX measures and implicit semantics (naming, formats, definitions)
- Linguistic model (synonyms, mapping business terms → fields)
- AI tooling (AI data schema, AI instructions, verified answers, Approved for Copilot)

An important architectural detail: for the “answer data questions” capability, Copilot requires that Power BI Q&A be enabled, because it uses the same underlying engine to build queries from natural language input.

2.2 Metadata, DAX, Relationships, and Descriptions

When you interact with Copilot in Power BI, Copilot does not directly query your database (SQL, lake, Databricks, etc.) as a generic query tool would. Instead, Copilot works with what Power BI provides as context (grounding) and with what can be resolved through the semantic model and/or the report.

In practice: Copilot does not have “free” access to data, but operates within the boundaries of what is:

- Published in the semantic model
- “Described” by metadata (schema, relationships, measures, definitions)
- Accessible to the user through permissions (RLS/OLS, workspace permissions, etc.)
- Relevant to the Copilot experience you are using (report pane, standalone, DAX query view, etc.)

Copilot “understands” and “calculates” only what can be expressed as:

- Selection of fields (columns/measures) already existing in the model
- Possibly ad hoc generated DAX
- Execution on the tabular engine of the semantic model
- Return of a result (text/visual)

If you ask for something that cannot be clearly mapped to the semantic model (or is not present), Copilot may:

- Ask for clarifications
- Respond incompletely
- Or (in some cases) “venture” into general knowledge not based on your data, with the risk of a plausible but unverifiable answer

2.3 What It Is Based On

Copilot in Power BI is based on Large Language Models provided through Azure OpenAI Service, belonging to the GPT family. The specific model used is neither exposed nor contractually specified, as Copilot is a managed service and can evolve over time. The role of LLM is not to directly query data, but to interpret natural language, map user intent to the semantic model, and orchestrate query generation and explainability. Calculations and data access always remain the responsibility of the Power BI engine, in compliance with security and governance policies.

Behind the scenes, a series of actions occurs:

- The LLM interprets natural language (prompt, follow-up)
- It does not directly query the data
- It is used to:
  - Understand intent
  - Map terms → metadata
  - Generate queries (DAX or instructions)
  - Explain results in natural language

The actual calculation always takes place in:

- Power BI / Fabric
- On the semantic model
- Via the tabular engine

What Composes the "Context"

The context that Power BI provides to Copilot varies based on the experience, but typically includes combinations of these elements:

A) Semantic Model Metadata

These are the "structural" information of the model, for example:

- Names of tables, columns, and measures
- Relationships and filter direction
- Formats and properties
- Descriptions (where present)
- Synonyms/linguistic model (Q&A)
- AI instructions, AI data schema, verified answers, Approved for Copilot (if configured)

Implication: if naming is ambiguous or measures are not "canonical," Copilot might choose the "wrong" fields even while generating formally correct DAX.

Let's consider an example discussing "sales." We could define a concrete measure:

[Sales Amount] = net revenue, excluding VAT, excluding canceled orders, including discounts according to rule X

While we might define a non-canonical measure:

[Sales], [Sales (raw)], [Revenue], [Net Sales], [Total Value], [Amount]

Or three different measures all called "Sales" in different folders or with similar naming.

In this second case, "sales" becomes an ambiguous term: Copilot may choose a measure the user did not intend.

A measure is typically "canonical" if it meets most of these criteria:

- Uniqueness: there is only one "official" measure for that KPI
- Explicit naming: clear name, with unit/semantics (Amount, %, Count) and no internal acronyms
- Documented definition: description explaining inclusions/exclusions and business rules
- Reuse: used in main reports and derived calculations (YoY, YTD, etc.)
- Governance: managed/approved (even informally) as a standard by the BI/Data team

"Canonical" does not mean "perfect" or "unchangeable." It means that, for the same concept, they are the source of truth, and all analyses should start there. Other measures may exist, but must be explicit variants (e.g., [Sales Amount - Gross], [Sales Amount - Including Returns]) and not compete semantically with the standard measure.

B) Report Metadata (When Copilot Is in the Report)

When you use Copilot in the report, Power BI can also include information about the report itself:

- Pages, titles, existing visuals
- Fields used in the visuals
- Page/report filters (note: these are not always automatically applied to responses, depending on the capability)

In this case, Copilot tends to favor fields already used in the report and to interpret questions in light of "what the report seems to say."

This means the same question can be implicitly disambiguated by the report, even if it isn't in the model strictly speaking.

Imagine having a report containing a "Sales Overview" page with a matrix for Region from DimStore and a card with [Sales Amount].

If I ask "Show me sales by area," Copilot is guided by this context and will most likely:

- Use [Sales Amount]
- Use DimStore[Region]

This behavior is a natural consequence of the design:

- Copilot uses all available context to resolve the user's intent
- The report is a strong additional semantic signal when present

Implication:

- Well-designed reports can "help" Copilot
- Ambiguous models work "better than expected" in reports
- If a model is to be used as an enterprise source for Copilot, it cannot rely on the context of individual reports for correct interpretation

Therefore, when Copilot is used within a report, Power BI can also provide the report's context (pages, visuals, fields already used), which serves as further semantic guidance.

When Copilot is used standalone or at the app level, this context does not exist and interpretation is based solely on the model and the prompt.

C) Your Prompt and the Conversation (Dynamic Context)

Copilot uses:

- Your question
- Any follow-ups
- Implicit specifications ("this month", "here", "above", etc.), which are often a source of ambiguity if not made explicit

A short or unconstrained prompt implicitly delegates key analytical decisions to Copilot—decisions that a human analyst would normally make explicitly. Every unspecified decision increases the space of possible interpretations, that is, semantic entropy.

In the context of Power BI, a prompt must be translated into a query (explicit or implicit). If constraints are missing, Copilot must "guess":

- Which measure to use (sales, margin, quantity, gross/net value)
- Which dimension to use (store region vs. customer region, category vs. subcategory)
- Which time period to consider (last month, YTD, entire history)
- Which granularity to adopt (day, month, year)
- Which implicit filters to apply or ignore (canceled orders, returns, channels)

Each possible choice is a fork in the road. The more forks you leave open, the higher the entropy and the more likely the result is:

- Formally correct
- Semantically plausible

- But different from what the user had in mind

An explicit prompt does not “help” Copilot in a generic sense: it narrows the problem Copilot needs to solve.

Practically, a good prompt:

- Reduces the number of candidate fields
- Constrains time choices
- Makes evident which measure is relevant
- Lowers the degree of freedom in query generation

This does not make Copilot “smarter,” but makes it more deterministic with respect to the user’s intent.

Naturally, it is important not to draw the wrong conclusion: “So just write long and verbose prompts.” That’s not the point.

A good semantic model already reduces entropy by itself, because it:

- Offers canonical measures
- Reduces naming ambiguities
- Encodes business rules

The prompt serves to complete the picture when:

- The analysis is exploratory
- The time or filter context is not “standard”
- The user wants a specific point of view

D) Output Data, Not Raw Data

Copilot does not typically receive “the whole dataset”: it receives the results of queries (or aggregations) needed to build the response. In some cases, the experience may include “data points” associated with the visuals/response (depending on the experience and report content).

Keep in mind that Copilot is not a lens that “sees everything,” but a system that obtains portions of result based on what it decides to request from the model.

A practical example to illustrate this:

If you ask: “Sales by region in the last 3 months,” Copilot must (simplifying):

- Understand what “sales” means → which measure? [Sales Amount]? [Net Revenue]?
- Understand what “region” means → DimStore[Region]? DimCustomer[Region]?
- Understand “last 3 months” → which date table? What is the “reference date”?
- Generate a consistent DAX query and execute it.

If the model does not clarify these concepts (naming, canonical measures, date table, AI instructions), Copilot may choose an interpretation different from what is expected. And this is where you understand why “Copilot does not see the database”: it sees a kind of semantic contract (your model) and makes decisions based on that.

2.3 Useful Mental Pipeline: Intent → Mapping → Query → Response

Without turning this paragraph into a whitepaper, the typical behavior in “data grounded” experiences is:

- Intent interpretation: the prompt is interpreted (what are you asking, at what granularity, with which filters)
- Semantic mapping: Copilot tries to map terms → fields/measures, mainly relying on names and linguistic model; the author can improve understanding with naming, best practices, and synonyms
- Query generation: may use existing measures and/or generate ad hoc DAX queries for calculations do not present in the model (e.g., YoY growth, ratio, set analysis)
- Execution: the tabular engine executes the query and produces a result set
- Rendering: Copilot builds a visual (and descriptive text) and attaches transparency elements like “How Copilot arrived at this”; the user can check the DAX and open DAX query view

This explains why “Copilot-friendly” first and foremost means model-friendly: if the mapping (step 2) is ambiguous or fragile, the generated query may be formally correct but not aligned with business definitions.

2.4 Intrinsic Limits: Semantic Ambiguity and Model Quality

Copilot exhibits two structural classes of limitations.

Ambiguity of natural language: “Sales” can mean Revenue, Net Sales, Gross Profit, GM%, Units, etc. If the model offers multiple plausible interpretations, Copilot may choose the “wrong” one for your organization (but still legitimate). The AI data schema tooling documentation uses an example where “sales” is interpreted as GPM and shows how narrowing the schema improves correctness.

Note that Copilot does not invent meanings.

It chooses among interpretations that the model itself makes plausible.

If the semantic model includes [Revenue], [Net Sales], [Gross Profit], [Gross Margin %], [Units Sold] and none of these is clearly declared as:

“This is the measure we mean when we say sales”

then the model is semantically ambiguous, not Copilot.

In practice: if the model allows for multiple coherent readings, Copilot will pick one, even if it is not the one culturally adopted in your company.

This is a fundamental distinction.

Frankly, before Copilot, the same problem already existed:

- a new analyst
- an external consultant
- a manager opening a report without context

could have:

    - used the wrong measure,
    - misinterpreted a KPI,
    - compared to non-comparable figures.

The difference is that before, the error was human and silent; now it is systemic and visible.

Copilot amplifies the semantics of the model:

- If the semantics are clear, it amplifies clarity.
- If it is ambiguous, it amplifies ambiguity.

This is uncomfortable, but also extremely useful—and necessary.

Quality of the semantic model: poor naming, complex relationships, missing measures, absent or unmarked date dimensions, etc., drastically increase the probability of misleading answers (even when the DAX is syntactically valid).

Copilot does not assess the "goodness" of the semantic model as a human expert would.

It assumes it as "truth."

This means: if a measure exists, Copilot considers it legitimate; if a relationship exists, Copilot uses it; if a time dimension is not clearly defined, Copilot must infer; if logic is not encoded in a measure, Copilot may generate one ad hoc.

The result can be:

- Syntactically correct DAX,
- Executable and performant query,
- Technically coherent visual,

But analysis semantically incorrect with respect to business rules not made explicit in the model.

Let's consider an example.

Fields with generic or technical names (Value, Amount, Flag, KPI1) do not provide sufficient semantic signals.

Copilot does not “understand” that:

- Amount = Net Sales
- Amount2 = Gross Sales
- Value = Profit

It just sees plausible alternatives.

If canonical measures are missing (YoY, YTD, Margin %, etc.), Copilot is encouraged to:

- build ad hoc calculations
- combine base measures in ways not aligned with company definitions

This is useful for exploration, but dangerous for shared KPIs.

Non-star schemas, many-to-many, bidirectional relationships or implicit filters make it harder to:

- understand which dimension is filtering what
- predict the outcome of an automatically generated query

An expert analyst “sees” these effects; Copilot applies them.

Copilot cannot compensate for a fragile semantic model.

In fact, it tends to expose its limitations more quickly than a traditional report.

This is not a flaw: it is a natural consequence of a system that generates queries dynamically instead of following predefined paths.

2.5 Conceptual Comparison: Copilot vs. Manual Query vs. Report Exploration

A useful comparison for decision makers and architects:

- **Manual query (DAX Studio / DAX query view / manually written measures)**: maximum precision and controllability; high cognitive cost; longer time
- **Traditional report + visual exploration:** robust, governable, repeatable; but limited to the paths anticipated by the author
- **Copilot:** accelerates unplanned exploratory analyses, prototyping activities, and contextual data queries, drastically reducing the time needed to get a first answer. However, speed does not eliminate the need for verification: generated responses must be validated through human control practices, such as inspecting the DAX used and verifying filters, periods, and metrics employed. For this approach to be

sustainable, a semantic model designed to minimize ambiguity is also needed, in which business definitions are explicit and key metrics are clearly coded.

3. Reference Semantic Model

Let's outline a simple yet coherent scenario for "sales analytics" designed to be Copilot-ready.

3.1 Model Objective

Support questions such as:

- Sales and margin by period/area/channel/category
- YoY comparison and trend
- Top/bottom N products and customers
- Consistent drill-down and slicing

3.2 Schema: Tables and Roles

Facts

- FactSales (granularity: sales line/document/item)
- SalesDateKey, ProductKey, CustomerKey, StoreKey, ChannelKey
- Quantity, NetAmount, CostAmount, DiscountAmount
- OrderStatus (to exclude cancelled), ReturnFlag

FactBudget (optional)

- DateKey, ProductKey/CategoryKey, StoreKey/RegionKey
- BudgetAmount

Dimensions

- DimDate (calendar table, with Date, Year, Quarter, Month, MonthName, Week, etc.)
- DimProduct (Product, Subcategory, Category, Brand)
- DimCustomer (Customer, Segment, Industry, Country/Region)
- DimStore (Store, City, Region, Area Manager)
- DimChannel (Online, Retail, Distributor, ...)

Relationships (star schema)

- DimDate[DateKey] 1→* FactSales[SalesDateKey] (single direction)
- DimProduct[ProductKey] 1→* FactSales[ProductKey]
- DimCustomer[CustomerKey] 1→* FactSales[CustomerKey]
- DimStore[StoreKey] 1→* FactSales[StoreKey]
- DimChannel[ChannelKey] 1→* FactSales[ChannelKey]

FactBudget has similar relationships (pay attention to granularity and ambiguity).

This structure reduces ambiguity: "by region" → DimStore[Region]; "by category" → DimProduct[Category]; "over time" → DimDate.

3.3 Key DAX Measures (with "business-first" naming)

Examples (with some simplification):

```
--Base measures

[Sales Amount] = SUM ( FactSales[NetAmount] )

[Sales Quantity] = SUM ( FactSales[Quantity] )

[Cost Amount] = SUM ( FactSales[CostAmount] )

[Gross Profit] = [Sales Amount] - [Cost Amount]

[Gross Margin %] = DIVIDE ( [Gross Profit], [Sales Amount] )

--Time intelligence (requires DimDate correctly marked as the Date table)

[Sales Amount LY] = CALCULATE ( [Sales Amount], SAMEPERIODLASTYEAR ( DimDate[Date] ) )

[Sales YoY %] = DIVIDE ( [Sales Amount] - [Sales Amount LY], [Sales Amount LY] )

[Sales YTD] = TOTALYTD ( [Sales Amount], DimDate[Date] )

[Sales YTD LY] = CALCULATE ( [Sales YTD], SAMEPERIODLASTYEAR ( DimDate[Date] ) )

[Sales YTD YoY %] = DIVIDE ( [Sales YTD] - [Sales YTD LY], [Sales YTD LY] )
```

Intentional choices:

- Explicit measures (not "implicit measures") with unique names and clear units
- Time intelligence ready to use (YoY, YTD) to reduce the need for "ad hoc calculations" in chat. (Copilot can create ad hoc DAX, but it's preferable to encode canonical metrics)
- Display folders (e.g., "Sales", "Profitability", "Time Intelligence") and consistent formats (currency, percentages)

3.4 Why This Model is "Copilot-ready"

By "Copilot-ready" I don't mean "magical": I mean it reduces the points where Copilot might misinterpret.

1. Consistent terminology: "Sales Amount" is not "Value" or "Total"; reduces mapping errors because Copilot relies heavily on field names when building visuals/responses

2. Single, strong date dimension: many Copilot errors come from "which date do you mean?" (OrderDate vs ShipDate vs InvoiceDate). A clear DimDate and disciplined use in measures reduce ambiguity
3. Simple relationships: star schema with single-direction filters reduces unwanted interpretations in calculations
4. AI tooling on the semantic model, not on the report:
    - AI data schema to narrow what Copilot has to reason about (for skills using it)
    - AI instructions to "teach" terminology and analytical rules
    - Verified answers for frequent/critical questions (curated, repeatable responses)
    - "Approved for Copilot" to reduce friction in Copilot standalone and, if desired, limit search to approved content

3.5 Concrete Example of AI Instructions (excerpt)

In the "Add AI instructions" tab, practical and verifiable instructions:

- For "Sales" we mean NetAmount (net revenue, net of discounts; excluding VAT).
- Always exclude OrderStatus = "Cancelled".
- The currency is EUR. Do not perform conversions.
- When the user asks for "by area", use DimStore[Region].
- When the user asks for "products", use DimProduct[ProductName] and allow drill on Category/Subcategory.

The AI instructions are saved on the model and consumed wherever Copilot interacts with that model; the end user does not see them and cannot disable them.

4. Examples of Correct Usage

The quality of Copilot depends on two factors: the **quality of the semantic model** and the **quality of the question** (prompt).

Let's analyze realistic scenarios and the reason why they yield correct results.

Scenario 1 — Guided and Verifiable YoY Analysis (Business User)

Prompt "Show me Sales Amount by Region for the last 3 months and the comparison with the same period last year. Highlight regions with Sales YoY % < -5%."

Why it Works

- It references measures and dimensions with **unique names** (Sales Amount, Region).
- It explicitly asks for the period and baseline (**last 3 months vs LY**), reducing temporal ambiguity.
- The **YoY metric is already in the model**, so Copilot mainly needs to select fields and filters, not "invent" logic.

How to Validate

- Open **"How Copilot arrived at this"** to verify the selected fields and filters.
- If Copilot generates ad hoc DAX, open the **DAX query view** for verification.

Scenario 2 — Top N with Geographic Filter and Relative Period

Prompt "Top 10 products by Sales Amount in Italy in the last month. Also display Gross Margin % for each one."

Why it Works

- **Top N is a pattern that Copilot handles well** when Product/Region/Date are clear fields.
- It is aligned with the **supported question types** (Top N filtered by region and time).

Operational Notes

- If the report already contains page filters (e.g., Italy), **do not assume that Copilot automatically applies them** within the pane: for this capability, it is explicit that it does not apply current report filters/slicers to the answers in the Copilot pane. Therefore, the prompt **must include** "Italy" and "last month."

Scenario 3 — "Set-based" Analysis (Customers who did not purchase)

Prompt "List the customers who purchased in Q2 2025 but not in Q3 2025. Show me the count by Segment."

Why it Works

- This is a typical case of **ad hoc calculation**: it requires a set difference logic that is often **not already in a measure**. The documentation indicates that Copilot can generate a DAX query for questions requiring ad hoc calculations ("Which customers didn't buy any products?").

How to Make it Enterprise-Safe

- If the scenario is frequent, **make it a certified measure/page**, don't leave it as "chat-only."

Scenario 4 — Using Verified Answers for "Business-Critical" Questions

Context In the semantic model, we created a verified answer "Current Month Sales KPI" with:

- card: Sales Amount MTD
- card: Sales YoY %
- trend: Sales Amount last 12 months
- filters "Available to users": Region, Channel, Category

Prompt (End-User) "Current Month Sales KPI for Online channel in North West."

Why it Works

- Copilot first searches for a match (exact or semantically similar) with the **trigger phrases of the verified answers**; if it finds a match, it returns the verified visual instead of generating a new answer.
- Verified answers are **stored at the semantic model level**; therefore, they apply to all reports that use that model.

Scenario 5 — Quick Creation of an "Executive Summary" Report Page

Prompt (in design mode) "Create an 'Executive Summary' page with:

- KPIs: Sales Amount, Gross Margin %, Sales YoY %;
- monthly trend Sales Amount (last 24 months);
- matrix Sales Amount by Region and Category;

- bar chart Top 10 products by Sales Amount."

Why it Works

- It clearly specifies desired objects, measures, granularity, time horizon, and visuals.
- Copilot supports the **creation and also the modification of pages** (add/change/delete visuals, undo/redo).

What NOT to Expect

- **No custom visuals** and **no styling/formatting changes** via Copilot.
- If the model has disabled implicit measures, Copilot cannot create report pages.

Scenario 6 — DAX Query View and Measure Descriptions (Model Design)

Correct Usage

- Ask Copilot to **propose a DAX query** or **explain a selected query**. Copilot uses the semantic model metadata and the selected query.
- **Generate measure descriptions**: Copilot uses the DAX formula and table name but does not read DAX comments or strings in double quotes within the formula.

5. Examples of Incorrect or Misleading Usage

Case 1 — Ambiguous Question: “Show me sales”

Prompt

“Show me sales by month.”

Problem

If the model contains measures like “Sales Amount,” “Gross Profit,” “Total GPM,” “Net Revenue,” etc., “sales” can be interpreted in multiple ways. The documentation specifies that in the presence of ambiguity, Copilot may choose a metric different from the team’s “standard” (e.g., GPM as an interpretation of sales).

Correction

- Make the prompt explicit: “Sales Amount”
- Reduce ambiguity using the AI data schema (exclude “distracting” measures) and/or AI instructions (“Sales = NetAmount”)

Case 2 — Incorrect Expectation: Asking “why” or “predict”

Prompt

“Why do sales drop every July?”“How many units will we sell next year?”

Problem

These requests imply anomaly detection and forecasting, which the “Ask Copilot for data from your model” capability does not support today.

Correction

- Rephrase in a “descriptive” way: “Show me sales by month for the last 3 years and compare July to the annual average”; then use human analysis and interpretation (or other dedicated features/services)

Case 3 — Relying on Current Page Filters/Slicers

Prompt

“In the current page context, show me Sales Amount by Category.”

Problem

For the “Ask Copilot for data from your model” capability, it is noted that Copilot does not apply the filters/slicers currently present on the page to the responses generated in the pane. This can result in a “global” answer when the user expects a “scoped” one.

Correction

- Explicitly state the filter in the prompt: “for Region = X, Channel = Y, period = …”, or use verified answers with available filters

Case 4 — Questions Outside the Data Scope → “General Knowledge” Response

Prompt

“What is the expected growth of the retail market in Italy and how will it impact our sales?”

Problem

Part of the question may not necessarily be in the semantic model; Copilot may answer using the LLM’s “general knowledge” if it cannot find grounding in the model. The risk is a plausible but unverifiable answer with company data.

Correction

- Separate the two questions: (1) internal data analysis; (2) external research (different processes, cited sources, different governance)

Case 5 — Unsupported Requests in Page Creation

Prompt

“Create a page with custom visual X and apply a specific style with corporate fonts and colors.”

Problem

Custom visuals and styling/formatting changes are not supported when creating/modifying pages through Copilot.

6. Best Practices for Designing “Copilot-Friendly” Semantic Models

The following best practices are intentionally concrete and applicable; they do not replace classic tabular best practices, but they emphasize areas where Copilot is most sensitive.

6.1 Naming: Reduce Linguistic Ambiguity Before Technical Ambiguity

Copilot uses field names as a strong indication for choosing what to use in visuals and responses.

- Measures: Use self-explanatory names with the unit implied in the name (e.g., “Sales Amount,” “Gross Margin %,” “Orders Count”).
- Columns: Avoid generic names like “Value,” “Amount,” or “Flag” without context. Prefer “Net Amount,” “Order Status,” “Return Flag.”
- Dates: Use DimDate[Date] as a unique reference; if you have multiple dates (Order/Ship/Invoice), name them explicitly and decide on a “primary” for standard analyses (and explain this in the AI instructions).

What to avoid:

- Internal abbreviations (e.g., “GPM,” “NR,” “Cst”) without synonyms/instructions
- Homonyms between tables and columns (e.g., “Region” in multiple tables without a clear hierarchy)

6.2 Model Structure: Simplify Mapping, Not Just Performance

- Prefer star schema and well-defined 1→* relationships.
- Limit many-to-many and bi-directional relationships where not necessary: these can be managed by experts but amplify ambiguity for a system that may have to “guess” intents.

6.3 Linguistic Modeling: Synonyms as a “Semantic Contract”

Copilot can be guided with synonyms for business-specific terms, improving its understanding of questions. As the documentation suggests, the author can guide Copilot with naming and synonyms, referring to the Q&A/linguistic schema tooling.

If you use Copilot to generate synonyms for Q&A, it is useful to know that (in that context) only the model’s metadata (table/field names) are sent to Copilot, not row contents or user questions.

In any case: synonyms must be reviewed and approved, not “accepted in bulk.”

6.4 Descriptions: Useful, But Know Where They Matter

Descriptions of tables/columns can provide context, but the documentation reminds us they are only used in DAX queries and Copilot search capabilities (not as a cure-all for every chat skill).

This implies a pragmatic choice:

- If your focus is “ask data questions,” invest first in AI data schema/instructions/verified answers.
- If your focus is “DAX query view” and semantic layer governance, descriptions are an important investment.

When Copilot:

- Generates a DAX query,
- Explains a selected query,
- Assists in the DAX Query View,

Descriptions become relevant because:

- They provide semantic context to interpret the role of a table or measure;
- They help Copilot explain “what a measure or part of the query does” in more readable terms.

In this scenario, descriptions enrich explanations, but:

- Do not change calculation logic,
- Do not correct wrong relationships or measures.

They are a cognitive support, not a primary tool for disambiguation.

In Copilot search experiences (standalone or at the app level), descriptions:

- Help Copilot understand what a semantic model or report is about,
- Improve matching between user requests and available content.

In this case, the description is used as a textual signal, similar to:

- Title,
- Name,
- High-level metadata.

6.5 “Prep Data for AI”: Use the Tools Correctly (and in the Right Order)

The set of features for “preparing data for AI” aims to reduce semantic ambiguity in the model and increase the likelihood that Copilot produces correct responses consistent with business definitions. The documentation explicitly notes that these features do not make Copilot deterministic: interaction remains probabilistic and sensitive to context and natural language. Their role is not to always guarantee the same output, but to narrow the space of possible interpretations and make results more stable, predictable, and manageable in real scenarios.

Recommended order:

1. AI data schema: Select a subset of tables/fields/measures for Copilot to reason over.

2. Verified answers: Frequently/critically asked questions with “human-approved” visual answers.
3. AI instructions: Rules for interpretation, terminology, table priorities.

AI data schema

Select “clean” fields with low ambiguity; remove confusing fields.

Note: the AI schema does not apply to all capabilities; for creating report pages, searching, or using DAX queries, Copilot requires the entire semantic model and does not consider the AI data schema.

AI instructions

Use them for definitions (“Sales = NetAmount”), rules (“busy season”), priorities among tables, and clarification requests (“if asking for sales by product, ask location”).

Important limits: Not visible or disable-able by end-users; maximum 10,000 characters.

Verified answers

Define these as curated answers for high-impact questions; Copilot returns them when it finds a match with a trigger phrase (even if semantically similar).

Consider filter limits: up to three filters “available to users,” slicers not supported, some date filters (relative date) not supported; there are quantity limits (e.g., 250 verified answers per model, 15 triggers per answer).

6.6 Mark “Approved for Copilot” and Manage Discoverability

Once you have prepared and validated the semantic model, you can mark it as Approved for Copilot. This flag does not change Copilot’s analytical behavior but serves a governance role: it signals that the model is a reliable and approved source for use in AI experiences. In Copilot standalone, content based on approved models is not subject to the “friction treatment” mechanisms provided for non-governed sources, making interaction smoother and more direct. Since approval is applied at the semantic model level, the status is automatically reflected on all reports using it, ensuring consistency in the Copilot experience regardless of the access point.

Friction treatment refers to mechanisms introduced by Copilot for responses based on content not explicitly approved. These mechanisms do not change calculations or data access but affect the user experience, introducing caution signals or usage limitations to distinguish governed sources from unvalidated ones. Marking a semantic model as Approved for Copilot removes this friction, treating the model as a reliable enterprise-level source.

Administratively, there is also a tenant setting to limit the standalone experience to searching only “Approved for Copilot” content.

This is a very concrete governance lever: it prevents Copilot from “pulling” from uncurated, sandbox, or legacy models with misaligned definitions.

6.7 Governance and Adoption: Enablement in Phases, Not “Switch On”

The documentation stresses two points:

- It is not enough to enable Copilot and expect automatic benefits; training, enablement, monitoring, and governance of usage are required.
- Enabling by security group and progressive rollout is a recommended practice.

One detail to be mindful of: content filtering can block LLM calls containing certain words/phrases; if these appear in your schema/metadata, Copilot can systematically fail. This is not a “theoretical edge case”: it is explicitly stated as a real risk.

7. Strategies for Those with Existing (Including Legacy) Models

In enterprise contexts, it is common to have “historic” semantic models featuring:

- technical naming conventions (columns like “VAL,” “DT,” “KPI1”)
- many redundant measures
- relationships that do not follow a star schema
- business rules that reside “in the heads” of just a few people

The goal here is not to “redo everything,” but to maximize the value-to-effort ratio.

7.1 First Constraint: Platform and Capability Prerequisites

First of all, ensure that Copilot can actually be used in your scenario:

- Copilot requires administrative enablement in Fabric/tenant settings
- regional capacity/support is needed: general requirements include at least Fabric F2 capacity
- trial SKUs/capacities are not supported; sovereign clouds are not supported

7.2 “Incremental” Strategy in 5 Steps

Step 1 — Reduce Ambiguity Without Touching the Physical Model: AI Data Schema

This is the most effective quick win because:

- it does not require DAX rewriting
- you can exclude “dangerous” fields/measures (ambiguous, duplicate, technical)
- it immediately improves answer quality in skills that use the schema

Step 2 — Encode Definitions: AI Instructions

Use these as an “operational glossary” and “analysis rules”:

- mapping business terms → correct fields
- prioritizing tables
- constant rules (exclude deleted, returns sign, currency)

Step 3 — Stabilize Recurring Questions: Verified Answers

If you have 10–20 questions that “deliver 80% of the value” (monthly KPIs, pipeline, top customer, etc.), verified answers:

- reduce the risk of variable interpretations
- make the experience repeatable and “trusted”

Step 4 — Set a Guardrail on Discoverability: Approved for Copilot + Standalone Limit

- Mark curated models as “Approved for Copilot”

- enable “only show approved items” in standalone if you want to prevent Copilot from pulling from uncurated legacy models

Step 5 — Targeted Structural Improvements

Only after the above quick wins, intervene where truly needed:

- Rename “canonical” measures (not all): Sales Amount, Gross Margin %, Orders Count
- Introduce a robust DimDate if missing (this will greatly impact the quality of time-based questions)
- Create standard YoY/YTD measures to avoid Copilot constantly generating ad hoc calculations

7.3 Criteria for Deciding What to Fix Immediately vs. Never

High priority (immediate impact on Copilot):

- ambiguous and duplicate terms (especially “sales,” “margin,” “cost”)
- absence of basic canonical measures
- absence/chaos of dates
- technical fields exposed to the user

Low priority (if Copilot is not the primary focus):

- deep refactoring of relationships if the schema is already “stable” and changing it is risky
- performance optimizations not related to the queries that Copilot generates

7.4 DevOps/ALM Note: LSDL, Git, and Deployment Pipelines

Changes to AI data schema and AI instructions are saved at the semantic model level, not in individual reports. Specifically, these configurations become part of the structural definition of the model (persisted in the internal LSDL format—Local Semantic Definition Language), following the model in all reports that use it and throughout its entire lifecycle.

8. The Right Mindset for Using Copilot

8.1 Copilot Changes the Interface, Not the Responsibility

- Copilot makes it easier to “ask” and “prototype,” but it does not eliminate:
- The responsibility to define metrics and semantics
- Validation of results and control of context (filters, granularity)
- Governance (which models are reliable, which are certified/approved)
- Data literacy among decision makers (knowing how to read and interpret a visual)

The documentation clearly states that Copilot does not replace those who create semantic models/reports and that a governed rollout with training and output evaluation is necessary.

8.2 Copilot as an “Amplifier”: Recommended Workflow

A practical workflow:

1. Ask with specific prompts (measure, dimension, period, filters)
2. Inspect “How Copilot arrived at this” and/or the generated DAX query
3. Compare with standard visuals or canonical measures (if they exist)
4. Industrialize: if the question is recurring, turn it into a measure/page/verified answer (don’t leave it in chat)
5. Govern: “Approved for Copilot,” standalone limitation, monitoring of capacity/usage

8.3 Prompting: Less “Prompt Engineering,” More “Semantic Engineering”

The value is not in finding the perfect phrase; it lies in designing a model where:

- Business concepts are unambiguous
- “Canonical” measures exist and have stable definitions
- Copilot has instructions/schema/verified answers so it doesn’t have to “guess”

9. Curiosities, Lesser-Known Details, and Practical Tips

This section gathers aspects that often emerge only in daily use and help avoid false positives/negatives.

9.1 “Clear Chat” Does Not Mean “Different Answer”

- “Clear chat” removes the conversational context, useful when changing the subject.
- In some cases, if the same prompt (or a semantically equivalent one) is posed on an unchanged semantic model within a limited time window, Copilot may reuse a previous answer instead of regenerating it. Clearing the chat resets the conversational context but does not guarantee invalidation of any service-level caching mechanisms. To get a different result, you must significantly change the prompt or modify the underlying model or data.

9.2 "Prep Data for AI" Lives on the Semantic Model, Not the Report

- All tooling updates are saved to the semantic model.
- Changes may take time to propagate; in Desktop, it's recommended to close and reopen the Copilot pane to see the effect, and in some cases, it could take minutes or even longer in complex scenarios.

9.3 AI Data Schema: Not "Universal"

- The AI data schema:
- Is invisible to consumers
- Applies only to capabilities that use it
- Is not considered in page creation, search, or DAX queries (where the entire model is needed)

Note that if the issue is "Copilot picks the wrong field when creating a report page," the AI data schema may not be the right lever; in that case, naming, relationships, measures, and (in part) instructions come into play.

9.5 Verified Answers: Attention to Filters and Maintenance

- Verified answers store persistent filters applied to the visual at setup; slicers are not supported.
- Some date filters (relative date) are not supported.
- They are model-level: changing the visual in the report does not automatically change the verified answer; to change it, use the management dialog (or recreate it).

9.6 "Ask Data Questions" (Model-Based)

- For the "Ask Copilot for data from your model" capability:
- Currently available in the service; Desktop support "coming soon"
- Supported language: English
- Does not apply current report filters/slicers to responses in the pane
- Does not support requests such as forecasting/anomaly detection

These details must be communicated to users; otherwise, Copilot may be considered "unreliable" for reasons of scope, not model quality.

9.7 Page Creation/Editing: Prerequisites and Blocks

- Q&A switch must be ON for report creation/editing
- Does not work with real-time streaming, live connection to Analysis Services, implicit measures disabled
- No custom visuals and no styling via Copilot
- This is an architectural point: if your governance requires implicit measures to be disabled for quality, accept that Copilot page generation is not a capability for that model (or consider separate "authoring-friendly" models).

9.8 Standalone Copilot: Automatic Workspace Selection and Capacity Implications

- The documentation describes that, in the absence of Fabric Copilot capacity (FCC), standalone can automatically select an eligible workspace for usage tracking/billing, with logic weighted by available capacity; the user can change it.
- For enterprise architecture: this is a strong reason to consider a dedicated FCC when usage grows.

9.9 Operations: Capacity Recognition and Rollout Times

- New capacities or scale-ups may require up to 24 hours before Copilot becomes available/recognized.
- Some features may also be rolled out progressively (e.g., in 2025, data preparation for AI was extended to the service with a global rollout).

9.10 Direction: Classic Q&A Deprecated, Copilot as an Alternative

- The Q&A experiences are scheduled for retirement in December 2026, with a recommendation to migrate to Copilot for natural language queries.
- This underscores what we've been writing: investing today in semantic quality (schema, synonyms, AI tooling) is not "optional," but a structural preparation for how users will interact with data in the future.

Where to start

If Copilot is already enabled (or about to be), the first step is not to “write better prompts,” but to look at the semantic model through a different lens. Tomorrow morning, pick a model that is genuinely used in production and ask yourself: do clearly canonical measures exist for the core KPIs? Do field names truly convey their business meaning, or do they require verbal explanation? If ambiguities emerge, that is the natural starting point.

The second step is to make explicit what is currently implicit. Use the Prepare data for AI tools not to “train the AI,” but to codify decisions that have already been made what we mean by sales, which date should be considered the reference date, which filters are always valid. Even a small number of well-written AI instructions and a minimal AI data schema can make an immediate difference in the quality of the answers.

The third step is to observe Copilot as a testing tool, not as an oracle. Use it to ask the kinds of questions that typically spark discussion between analysts and stakeholders. If Copilot produces an unexpected answer, this is not a failure; it is a signal. It is surfacing an ambiguity in the model that is worth addressing before it produces downstream effects.

Finally, invest time in targeted enablement, not generic training. Teach key users how to ask questions, how to interpret the answers, how to validate them, and when to trust them. Teach those who build semantic models and those who develop reports how to design with Copilot in mind. This is the step that transforms Copilot from an enabled feature into a capability that is truly adopted.

Starting from here does not require a revolution, but a shift in perspective: treating Copilot not as an isolated experiment, but as a mirror of the maturity of one’s data ecosystem. And it is precisely from this first, small step that the most durable value emerges.

Conclusion

Copilot in Power BI represents an important step forward in the way organizations can access, explore, and understand their data. Not because it replaces human work, but because it makes analytical capabilities faster and more widely accessible—capabilities that once required specialized skills and longer timeframes. When embedded in a solid context, Copilot can become a true accelerator of insight, collaboration, and decision quality.

The value of Copilot, however, does not lie in the prompt or in the interface, but in the quality of the context in which it operates. A clear, governed, and consistent semantic model turns Copilot into a reliable tool; an ambiguous model, instead, amplifies its fragilities. In this sense, Copilot is not only a new feature, but also a powerful catalyst for maturity: it makes architectural choices, business definitions, and governance practices—often previously implicit—explicit and visible.

Copilot is a multiplier of the quality of your semantic model; it is not a substitute.

A conscious adoption of Copilot does not require a radical rewrite of existing systems, but a progressive journey: clarifying core metrics, reducing semantic ambiguity, investing in training and validation, and leveraging the governance tools provided by the platform. This is an investment that pays off over time, because it improves not only the Copilot experience, but the overall quality of the data ecosystem.

Viewed from this perspective, Copilot is not a shortcut, but a multiplier. It does not remove the need to understand data; it makes that understanding more accessible, shared, and verifiable. It is precisely in this balance between automation and responsibility that Copilot's most durable value resides.