Power BI Copilot

Guida pratica a ciò che Copilot vede, capisce e non può sapere

Andrea Benedetti
Sr Solution Engineer Data & AI, Microsoft
https://www.linkedin.com/in/abenedetti/

Power BI Copilot

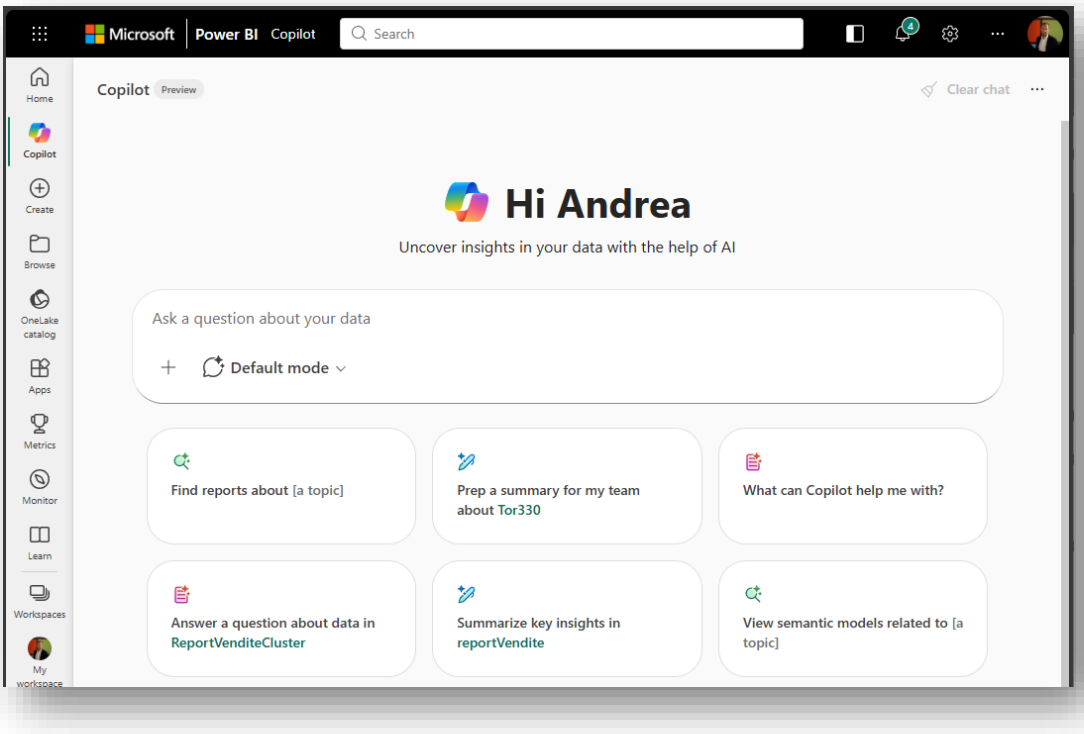
Guida pratica a ciò che Copilot vede, capisce e non può sapere

Dec 2025

Introduzione

Questo documento ha l'obiettivo di fornire una lettura chiara, tecnica e senza filtri sull'utilizzo di Copilot in Power BI, aiutando professionisti Data & Analytics, architetti e decision maker a comprenderne le reali capacità, i limiti strutturali e le condizioni necessarie perché produca valore concreto nel contesto enterprise.

Copilot rappresenta un'evoluzione significativa nel modo di interagire con i dati: abbassa la barriera di accesso all'esplorazione, accelera analisi non pianificate, prototipazione e interrogazioni contestuali. Tuttavia, il suo valore non è automatico. Abilitare Copilot non equivale ad adottarlo con successo, e Copilot non "funziona da solo". Come ogni capability avanzata, richiede un contesto adeguato per esprimere il proprio potenziale.



In particolare, Copilot presuppone la presenza di modelli semantici solidi, definizioni di business esplicite, competenze diffuse, pratiche di validazione, formazione degli utenti e un chiaro framework di governance. In assenza di questi elementi, il rischio non è tanto tecnologico quanto operativo: risposte instabili, interpretazioni ambigue e una percezione di affidabilità non sempre allineata alla realtà dei dati. In questo senso, Copilot non è "gratis": il suo costo non è solo computazionale, ma anche organizzativo, culturale e architetturale.

Esiste inoltre una differenza sostanziale rispetto al passato. Prima dell'introduzione di Copilot, ambiguità semantiche ed errori di interpretazione erano prevalentemente umani e spesso

silenziosi, legati all'esperienza individuale di chi costruiva o consumava i report. Con Copilot, queste ambiguità diventano visibili. Copilot non introduce nuove incertezze, ma amplifica la semantica del modello su cui opera: quando la semantica è chiara e ben codificata, amplifica chiarezza e coerenza; quando è ambigua o incompleta, amplifica quella stessa ambiguità. Questo effetto può risultare scomodo, ma è anche estremamente utile, perché rende evidenti limiti e incoerenze che prima rimanevano nascosti, trasformando Copilot in un potente strumento di stress test del semantic model.

Questo documento nasce con l'obiettivo di rendere esplicite queste dinamiche e di fornire una guida concreta su come progettare, utilizzare e governare Copilot in Power BI in modo consapevole. Per metterlo nelle condizioni di esprimere valore reale, affidabile e sostenibile nel tempo.

1. Cos'è Copilot in Power BI (e cosa NON è)

1.1 Definizione corretta

Copilot in Power BI non è una singola funzionalità, ma un insieme di esperienze ("Copilot experiences") integrate in Power BI/Fabric che usano generative AI per:

- **consumare** contenuti (riassunti, Q&A guidato, visual ad hoc) a partire da report e modelli semantici;
- **creare** e **modificare** contenuti (pagine report, narrative, DAX, descrizioni misure);
- **guidare** l'utente a trovare contenuti (standalone/app-scoped) e a interagire con i dati in linguaggio naturale.

A livello di "forma", oggi è utile pensare a tre contenitori principali:

- **Copilot pane (report agent)**: il riquadro laterale dentro un report
- **Standalone Copilot (Power BI agent)**: esperienza full-screen che trova/analizza qualunque item a cui l'utente ha accesso
- **Copilot in apps (app agent)**: esperienza full-screen "scoped" al contenuto di un'app Power BI

1.2 Cosa può fare oggi (in modo pratico)

Dal punto di vista di un team Data & Analytics, Copilot in Power BI non va letto come un insieme di funzionalità isolate, ma come un insieme coerente di capacità che agiscono su piani diversi dell'esperienza analitica.

Il valore si concentra in tre aree operative:

1. **Chat con i dati** (business users e utenti avanzati)
    - "Answer questions about the data"
    - accesso a trasparenza/validazione tramite "How Copilot arrived at this"
    - possibilità di aggiungere visual generati alla pagina
2. **Supporto allo sviluppo di report**
    - creazione e modifica di pagine report via prompt (aggiungi/cambia/cancella visual, undo/redo)
    - generazione di narrative summary

3. **Supporto allo sviluppo del modello/semantic layer**
    - scrittura/spiegazione di DAX query (DAX query view) e creazione descrizioni misure
    - "prep" del modello per AI tramite AI data schema, AI instructions, verified answers, e "Approved for Copilot"

1.3 Cosa NON fa (e perché)

Copilot non è un "analista automatico" che genera insight avanzati a richiesta.
Per esempio, nella capability "Ask Copilot for data from your model" è esplicito che non risponde (oggi) a richieste che implicano **anomaly detection**, **forecasting** o "key influencers"; sono esempi di domande non supportate ("Why…?", "How many … next year?").

Copilot non è un sostituto delle persone che modellano e costruiscono report.
La documentazione lo afferma senza ambiguità: Copilot mira a aumentare efficienza e qualità, non a sostituire chi costruisce semantic model e report; i risultati generati non rimpiazzano la comprensione di contesto e problema che un analista/BI developer ha.

Copilot non "capisce i dati" oltre ciò che il modello semantico gli consente di capire.
Se la domanda è "related to data in the semantic model", Copilot usa il semantic model.

2. Architettura concettuale: cosa succede “dietro le quinte”

2.1 Il modello semantico è il centro di tutto

In Power BI, Copilot “ragiona” soprattutto attraverso ciò che il semantic model espone:

- **schema** (tabelle, colonne, misure, gerarchie, proprietà)
- **relazioni** e direzionalità
- **misure DAX** e semantica implicita (naming, formati, definizioni)
- **linguistic model** (sinonimi, mapping termini business → campi)
- **AI tooling** (AI data schema, AI instructions, verified answers, Approved for Copilot)

Un dettaglio architetturale importante: per la capability di “answer data questions” Copilot richiede che **Power BI Q&A sia abilitato**, perché usa lo stesso engine sottostante per costruire query a partire dall’input naturale.

2.2 Metadata, DAX, relazioni e descrizioni

Quando interagisci con Copilot in Power BI, Copilot non interroga direttamente il tuo database (SQL, lake, Databricks, ecc.) come farebbe un tool di query generico. Copilot lavora invece su ciò che Power BI gli mette a disposizione come contesto (grounding) e su ciò che è risolvibile tramite il modello semantico e/o il report.

In pratica: Copilot non ha accesso “libero” ai dati, ma opera dentro i confini di ciò che:

1. è pubblicato nel semantic model
2. è “descritto” dai metadati (schema, relazioni, misure, definizioni)
3. è accessibile all’utente tramite le autorizzazioni (RLS/OLS, permessi di workspace, ecc.)
4. è rilevante rispetto all’esperienza Copilot che stai usando (pane nel report, standalone, DAX query view, ecc.)

Copilot “capisce” e “calcola” solo ciò che può essere espresso come:

- selezione di campi (colonne/misure) già esistenti nel modello
- eventualmente DAX generato ad hoc
- esecuzione sul motore tabular del semantic model
- restituzione di un risultato (testo/visual)

Se invece chiedi qualcosa che non è chiaramente mappabile al semantic model (o non è presente), Copilot può:

- chiedere chiarimenti
- rispondere in modo incompleto
- oppure (in alcuni casi) "sconfinare" su conoscenza generale non basata sui tuoi dati, con rischio di risposta plausibile ma non verificabile

2.3 Su cosa si basa

Copilot in Power BI si basa su Large Language Models forniti tramite Azure OpenAI Service, appartenenti alla famiglia GPT. Il modello specifico utilizzato non è esposto né contrattualizzato, poiché Copilot è un servizio gestito e può evolvere nel tempo. Il ruolo dell'LLM non è quello di interrogare direttamente i dati, ma di interpretare il linguaggio naturale, mappare l'intento dell'utente sul semantic model e orchestrare la generazione di query ed explainability. I calcoli e l'accesso ai dati restano sempre demandati al motore Power BI, nel rispetto delle policy di sicurezza e governance.

Dietro le quinte avvengono una serie di cose:

- l'LLM interpreta il linguaggio naturale (prompt, follow-up)
- non interroga direttamente i dati
- viene usato per:
    - capire l'intento
    - mappare termini → metadati
    - generare query (DAX o istruzioni)
    - spiegare risultati in linguaggio naturale

Il calcolo vero avviene sempre in:

- Power BI / Fabric
- sul semantic model
- tramite il motore tabular

Che cosa compone il “contesto”

Il contesto che Power BI fornisce a Copilot varia in base all’esperienza, ma tipicamente include combinazioni di questi elementi:

A) Metadati del semantic model

Sono le informazioni “strutturali” del modello, ad esempio:

- nomi di tabelle, colonne e misure
- relazioni e direzione dei filtri
- formati e proprietà
- descrizioni (dove presenti)
- sinonimi/linguistic model (Q&A)
- AI instructions, AI data schema, verified answers, Approved for Copilot (se configurati).

Implicazione: se il naming è ambiguo o le misure non sono “canoniche”, Copilot potrebbe scegliere campi “sbagliati” pur generando DAX formalmente corretto.

Proviamo a fare un esempio parlando di “vendite”.

Potremmo definire una misura concreta:

- [Sales Amount] = ricavi netti, IVA esclusa, esclusi ordini cancellati, inclusi sconti secondo regola X

Mentre potremmo definire una misura non canonica:

- [Sales], [Sales (raw)], [Revenue], [Net Sales], [Total Value], [Amount]
- oppure 3 misure diverse tutte chiamate “Sales” in folder diversi o con naming simile

In questo secondo caso, “vendite” diventa un termine ambiguo: Copilot può scegliere una misura che l’utente non intendeva.

Una misura è tipicamente “canonica” se soddisfa quasi tutti questi criteri:

- Unicità: per quel KPI esiste una sola misura “ufficiale”
- Naming esplicito: nome chiaro, con unità/semantica (Amount, %, Count) e senza acronimi interni
- Definizione documentata: descrizione che spiega inclusioni/esclusioni e regole business
- Riutilizzo: è usata nei report principali e nei calcoli derivati (YoY, YTD, ecc.)

- Governance: è gestita/approvata (anche informalmente) come standard dal team BI/Data

“Canoniche” non significa “perfette” o “immutabili”. Significa che, a parità di concetto, sono la fonte di verità e tutte le analisi dovrebbero partire da lì. Le altre misure possono esistere, ma devono essere dichiaratamente varianti (es. [Sales Amount - Gross], [Sales Amount - Including Returns]) e non competere semanticamente con la misura standard.

B) Metadati del report (quando Copilot è nel report)

Quando usi Copilot nel report, Power BI può includere anche informazioni sul report stesso:

- pagine, titoli, visual presenti
- campi usati nei visual
- filtri di pagina/report (nota: non sempre vengono applicati automaticamente alle risposte, dipende dalla capability)

In questo caso, Copilot tende a privilegiare i campi già usati nel report e a interpretare le domande alla luce di “quello che il report sembra dire”.

Questo significa che la stessa domanda può essere implicitamente disambiguata dal report, anche se non lo è nel modello in senso stretto.

Immaginiamo di avere un report che contiene una pagina “Sales Overview” con una matrice per Region da DimStore e una card con [Sales Amount].

Se chiedo “Mostrami le vendite per area”, Copilot è guidato da questo contesto e molto probabilmente:

- userà [Sales Amount]
- userà DimStore[Region]

Questo comportamento è una conseguenza naturale del design:

- Copilot usa **tutto il contesto disponibile** per risolvere l’intento dell’utente
- il report è un *forte segnale semantico* aggiuntivo, quando presente

Implica che:

- report ben progettati possono “aiutare” Copilot
- modelli ambigui funzionano “meglio del previsto” nei report
- se un modello deve essere usato come **fonte enterprise per Copilot**, non può affidarsi al contesto dei singoli report per essere interpretato correttamente

Quindi, quando Copilot è usato all’interno di un report, Power BI può fornirgli anche il contesto del report stesso (pagine, visual, campi già utilizzati), che funge da ulteriore guida semantica.

Quando invece Copilot è usato in modalità standalone o a livello di app, questo contesto non esiste e l’interpretazione si basa esclusivamente sul modello e sul prompt.

C) Il tuo prompt e la conversazione (contesto “dinamico”)

Copilot usa:

- la tua domanda
- eventuali follow-up
- le specifiche implicite (“questo mese”, “qui”, “in alto”, ecc.), che però sono spesso fonte di ambiguità se non esplicitate

Quando un utente scrive un prompt molto breve o poco vincolante, sta implicitamente delegando a Copilot molte decisioni che altrimenti prenderebbe un analista umano. Ogni decisione non esplicitata aumenta lo spazio delle interpretazioni possibili, cioè l’entropia semantica.

Nel contesto di Power BI, un prompt deve essere tradotto in una query (esplicita o implicita). Se mancano vincoli, Copilot deve “indovinare”:

- **quale misura** usare (vendite, margine, quantità, valore lordo/netto)
- **quale dimensione** usare (regione store vs regione cliente, categoria vs sottocategoria)
- **quale periodo temporale** considerare (ultimo mese, YTD, intera storia)
- **quale granularità** adottare (giorno, mese, anno)
- **quali filtri impliciti** applicare o ignorare (ordini cancellati, resi, canali)

Ogni scelta possibile è un bivio. Più bivi lasci aperti, più aumenta l’entropia e più è probabile che il risultato sia:

- formalmente corretto
- semanticamente plausibile
- ma diverso da ciò che l'utente aveva in mente

Un prompt esplicito non "aiuta" Copilot in senso generico: **restringe il problema** che Copilot deve risolvere.

Da un punto di vista pratico, un buon prompt:

- riduce il numero di campi candidati
- vincola le scelte temporali
- rende evidente quale misura è rilevante
- abbassa il grado di libertà nella generazione della query

Questo non rende Copilot più "intelligente", ma lo rende **più deterministico** rispetto all'intento dell'utente.

Naturalmente è importante non trarre la conclusione sbagliata: "Allora basta scrivere prompt prolissi e lunghissimi." Non è questo il punto.

Un buon modello semantico riduce già di per sé l'entropia, perché:

- offre misure canoniche
- riduce le ambiguità di naming
- codifica regole business

Il prompt serve a completare il quadro quando:

- l'analisi è esplorativa
- il contesto temporale o di filtro non è "standard"
- l'utente vuole un punto di vista specifico

D) Dati di output, non dati grezzi

Copilot non riceve tipicamente “tutto il dataset”: riceve i risultati di query (o aggregazioni) necessari a costruire la risposta. In alcuni casi l’esperienza può includere “data points” associati ai visual/risposta (dipende dall’esperienza e dal contenuto del report).

Va tenuto a mente che Copilot non è una lente che “vede tutto”, ma un sistema che ottiene porzioni di risultato in base a ciò che decide di chiedere al modello.

Un esempio pratico che rende l’idea

Se chiedi: “Vendite per regione negli ultimi 3 mesi”, Copilot deve fare (semplificando):

1. Capire cosa significa “vendite” → quale misura? [Sales Amount]? [Net Revenue]?
2. Capire cosa significa “regione” → DimStore[Region]? DimCustomer[Region]?
3. Capire “ultimi 3 mesi” → quale tabella date? qual è la “data di riferimento”?
4. Generare una DAX query coerente e farla eseguire.

Se il modello non chiarisce questi concetti (naming, misure canoniche, date table, AI instructions), Copilot può scegliere un’interpretazione diversa da quella attesa. E qui si comprende bene perché “Copilot non vede il database”: vede una sorta di contratto semantico (il tuo modello) e prende decisioni su quello.

2.3 Pipeline mentale utile: intent → mapping → query → risposta

Senza trasformare questo paragrafo in un whitepaper, il comportamento tipico nelle esperienze “data grounded” è:

1. **Interpretazione dell’intento**: il prompt viene interpretato (cosa stai chiedendo, a che granularità, con quali filtri)
2. **Mapping semantico**: Copilot prova a mappare termini → campi/misure, basandosi soprattutto su nomi e linguistic model; l’autore può migliorare la comprensione con naming, best practices e sinonimi
3. **Generazione di query**: può usare misure esistenti e/o generare **DAX query ad hoc** per calcoli non presenti nel modello (es. YoY growth, ratio, set analysis)
4. **Esecuzione**: il motore tabular esegue la query e produce un resultset

5. **Rendering**: Copilot costruisce un visual (e testo descrittivo) e allega elementi di trasparenza come “How Copilot arrived at this”; l’utente può verificare il DAX e aprire DAX query view

Questo spiega perché “Copilot-friendly” significa innanzitutto **model-friendly**: se il mapping (passo 2) è ambiguo o fragile, la query generata può risultare formalmente corretta ma semanticamente non allineata alle definizioni di business.

2.4 Limiti intrinseci: ambiguità semantica e qualità del modello

Copilot soffre (potenzialmente) di due classi di limiti strutturali:

- **Ambiguità del linguaggio naturale**: “vendite” può significare Revenue, Net Sales, Gross Profit, GM%, Units, ecc. Se il modello offre più interpretazioni plausibili, Copilot può scegliere quella “sbagliata” per la vostra organizzazione (ma comunque legittima). La stessa documentazione del tooling AI data schema usa un esempio di “sales” interpretato come GPM e mostra come restringere lo schema migliori la correttezza.

  Attenzione che Copilot **non inventa significati**.
  Sceglie tra **interpretazioni che il modello stesso rende plausibili**.

  Se nel semantic model convivono: [Revenue], [Net Sales], [Gross Profit], [Gross Margin %], [Units Sold] e nessuna di queste è chiaramente dichiarata come:

  “questa è *la* misura che intendiamo quando diciamo *vendite*”

  allora **il modello è semanticamente ambiguo**, non Copilot.

  Nella pratica: se il modello consente più letture coerenti, Copilot ne sceglierà una, anche se non è quella culturalmente adottata nella vostra azienda.

  Questa è una distinzione fondamentale.

  In tutta franchezza, prima di Copilot, lo stesso problema esisteva già:

  - un nuovo analista
  - un consulente esterno
  - un manager che apre un report senza contesto

  avrebbero potuto:

  - usare la misura sbagliata,
  - interpretare male un KPI,
  - confrontare grandezze non confrontabili.

La differenza è che: **prima l'errore era umano e silenzioso ora è sistemico e visibile.**

Copilot amplifica la semantica del modello.
Se la semantica è chiara, amplifica chiarezza.
Se è ambigua, amplifica ambiguità.

A mio avviso questo è scomodo, ma è anche estremamente utile.

- **Qualità del semantic model**: naming povero, relazioni complesse, misure mancanti, date dimension assente o non marcata, ecc. aumentano drasticamente la probabilità di risposte fuorvianti (anche quando la DAX è sintatticamente valida).

  Copilot non valuta la "bontà" del modello semantico come farebbe un esperto umano. Lo assume come "verità".

  Questo significa che: se una misura esiste, Copilot la considera legittima; se una relazione esiste, Copilot la usa; se una dimensione temporale non è chiaramente definita, Copilot deve inferire; se una logica non è codificata in una misura, Copilot può generarne una ad hoc.

  Il risultato può essere:

  - DAX sintatticamente corretto,
  - query eseguibile e performante,
  - visual coerente dal punto di vista tecnico,

  ma analisi semanticamente scorretta rispetto alle regole di business non esplicitate nel modello.

  Proviamo a fare un esempio.

  Campi con nomi generici o tecnici (Value, Amount, Flag, KPI1) non forniscono segnali semantici sufficienti.

  Copilot non "capisce" che:

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

  vede solo alternative plausibili.

Se mancano misure canoniche (YoY, YTD, Margin %, ecc.), Copilot è incentivato a:

- costruire calcoli ad hoc
- combinare misure base in modi non allineati alle definizioni aziendali

Questo è utile per l'esplorazione, ma pericoloso per KPI condivisi.

Schemi non a stella, many-to-many, relazioni bidirezionali o filtri impliciti rendono più difficile:

- capire quale dimensione stia filtrando cosa
- prevedere il risultato di una query generata automaticamente

Un analista esperto "vede" questi effetti; Copilot li applica.

Copilot non può compensare un semantic model fragile.
Anzi, tende a esporne i limiti più rapidamente rispetto a un report tradizionale.

Questo non è un difetto: è una conseguenza naturale di un sistema che genera query dinamicamente invece di seguire percorsi predefiniti.

2.5 Confronto concettuale: Copilot vs query manuale vs report exploration

Un confronto utile per decision maker e architetti:

- **Query manuale (DAX Studio / DAX query view / misure scritte a mano)**: massima precisione e controllabilità; costo cognitivo alto; tempo più lungo
- **Report tradizionale + esplorazione visuale**: robusto, governabile, ripetibile; però limitato ai percorsi previsti dall'autore
- **Copilot**: accelera analisi esplorative non pianificate, attività di prototipazione e interrogazioni contestuali sui dati, riducendo drasticamente il tempo necessario per ottenere una prima risposta. Questa velocità, però, non elimina la necessità di verifica: le risposte generate devono essere validate attraverso pratiche di controllo umano, come l'ispezione del DAX utilizzato e la verifica di filtri, periodi e metriche impiegate. Perché questo approccio sia sostenibile, è inoltre necessario un semantic model progettato per minimizzare l'ambiguità, in cui le definizioni di business siano esplicite e le metriche principali chiaramente codificate.

3. Modello semantico di riferimento

Proviamo a delineare uno scenario semplice, ma coerente, di “sales analytics” pensato per essere Copilot-ready.

3.1 Obiettivo del modello

Supportare domande come:

- vendite e margine per periodo/area/canale/categoria
- confronto YoY e trend
- top/bottom N prodotti e clienti
- drill-down e slicing coerente

3.2 Schema: tabelle e ruolo

Fatti

- **FactSales** (granularità: riga di vendita / documento / item)
    - SalesDateKey, ProductKey, CustomerKey, StoreKey, ChannelKey
    - Quantity, NetAmount, CostAmount, DiscountAmount
    - OrderStatus (per escludere annullati), ReturnFlag
- **FactBudget** (opzionale)
    - DateKey, ProductKey/CategoryKey, StoreKey/RegionKey
    - BudgetAmount

Dimensioni

- **DimDate** (tabella calendario, con Date, Year, Quarter, Month, MonthName, Week, ecc.)
- **DimProduct** (Product, Subcategory, Category, Brand)
- **DimCustomer** (Customer, Segment, Industry, Country/Region)
- **DimStore** (Store, City, Region, Area Manager)
- **DimChannel** (Online, Retail, Distributor, ...)

Relazioni (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 con relazioni analoghe (attenzione a granularità e ambiguità).

Questa struttura riduce ambiguità: “per regione” → DimStore[Region]; “per categoria” → DimProduct[Category]; “nel tempo” → DimDate.

3.3 Misure DAX chiave (con naming “business-first”)

Esempi (mi passerete la semplificazione):

```
-- 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 (richiede DimDate corretta e marcata come 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] )
```

Scelte intenzionali:

- **Misure esplicite** (non “implicit measures”) con nomi univoci e unità chiare

- **Time intelligence** pronta all'uso (YoY, YTD) per ridurre la necessità di "ad hoc calculations" nella chat. (Copilot può creare DAX ad hoc, ma è preferibile codificare le metriche canoniche)
- **Display folders** (es. "Sales", "Profitability", "Time Intelligence") e formati coerenti (valuta, percentuali)

3.4 Perché questo modello è "Copilot-ready"

Per "Copilot-ready" non intendo "magico": intendo che riduce i punti in cui Copilot può equivocare.

1. **Terminologia coerente**: "Sales Amount" non è "Value" o "Total"; riduce mapping errati perché Copilot si basa molto sui nomi campo quando costruisce visual/risposte
2. **Dimensione data unica e forte**: gran parte degli errori Copilot nasce da "quale data intendi?" (OrderDate vs ShipDate vs InvoiceDate). Una DimDate chiara e l'uso disciplinato in misure riduce ambiguità
3. **Relazioni semplici**: star schema con filtri unidirezionali riduce interpretazioni indesiderate nei calcoli.
4. **AI tooling** sul semantic model, non sul report:
    - AI data schema per restringere ciò su cui Copilot deve ragionare (per le skill che lo usano)
    - AI instructions per "insegnare" terminologia e regole analitiche
    - Verified answers per domande frequenti/critiche (risposte curate, ripetibili)
    - "Approved for Copilot" per ridurre frizioni nel Copilot standalone e, se desiderato, limitare la ricerca a contenuti approvati

3.5 Esempio concreto di AI instructions (estratto)

Nel tab "Add AI instructions", istruzioni pratiche e verificabili:

- "Per 'Sales' intendiamo **NetAmount** (ricavi netti, al netto di sconti; IVA esclusa)."
- "Escludi sempre OrderStatus = 'Cancelled'."
- "La valuta è EUR. Non fare conversioni."
- "Quando l'utente chiede 'per area', usa DimStore[Region]."

- “Quando l’utente chiede ‘prodotti’, usa DimProduct[ProductName] e consenti drill su Category/Subcategory.”

Le AI instructions sono salvate sul modello e consumate ovunque Copilot interagisca con quel modello; l’utente finale non le vede e non può disabilitarle.

4. Esempi di utilizzo corretto

La qualità di Copilot dipende da due fattori: **qualità del semantic model** e **qualità della domanda**.

Proviamo ad analizzare scenari realistici e il motivo del perché diano risultati corretti.

Scenario 1 — Analisi YoY guidata e verificabile (business user)

Prompt
"Mostrami *Sales Amount* per *Region* negli ultimi 3 mesi e il confronto con lo stesso periodo dell'anno precedente. Evidenzia le regioni con *Sales YoY %* < -5%."

Perché funziona

- Referenzia misure e dimensioni con nomi univoci (Sales Amount, Region)
- Chiede esplicitamente periodo e baseline (ultimi 3 mesi vs LY), riducendo ambiguità temporale
- La metrica YoY è già nel modello, quindi Copilot deve soprattutto selezionare campi e filtri, non "inventare" logica

Come validare

- Aprire "How Copilot arrived at this" per verificare campi e filtri scelti
- Se Copilot genera DAX ad hoc, aprire DAX query view per verifica

Scenario 2 — Top N con filtro geografico e periodo relativo

Prompt
"Top 10 prodotti per *Sales Amount* in Italia nell'ultimo mese. Visualizza anche *Gross Margin %* per ciascuno."

Perché funziona

- Top N è un pattern che Copilot gestisce bene quando Product/Region/Date sono campi chiari
- È allineato ai question types supportati (top N filtrato per regione e tempo)

Note operative

- Se il report contiene già filtri pagina (es. Italia), non dare per scontato che Copilot li applichi automaticamente dentro il pane: per questa capability è esplicito che non applica filtri/slicer correnti del report alle risposte nel Copilot pane. Quindi il prompt deve includere “Italia” e “ultimo mese”

Scenario 3 — Analisi “set-based” (clienti che non hanno acquistato)

Prompt
“Elenca i clienti che hanno acquistato nel Q2 2025 ma non nel Q3 2025. Mostrami il conteggio per *Segment*.”

Perché funziona

- È un caso tipico di **ad hoc calculation**: richiede una logica di set difference che spesso non è già in misura. La documentazione indica che Copilot può generare DAX query per domande che richiedono calcoli ad hoc (“Which customers didn’t buy any products?”)

Come renderlo enterprise-safe

- Se lo scenario è frequente, farlo diventare una misura/una pagina certificata, non lasciarlo come “chat-only”

Scenario 4 — Uso di Verified Answers per domande “business-critical”

Contesto
Nel semantic model abbiamo creato un verified answer “KPI vendite mese corrente” con:

- card: Sales Amount MTD
- card: Sales YoY %
- trend: Sales Amount ultimi 12 mesi
- filtri “Available to users”: Region, Channel, Category

Prompt (utente finale)
“KPI vendite mese corrente per canale Online in Nord Ovest.”

Perché funziona

- Copilot prima cerca una corrispondenza (esatta o semanticamente simile) con i trigger phrase dei verified answers; se trova match, restituisce il visual verificato invece di generare una risposta nuova

- I verified answers sono memorizzati a livello di semantic model; quindi, valgono per tutti i report che usano quel modello

Scenario 5 — Creazione rapida di una pagina report "Executive Summary"

Prompt (in design mode)
"Crea una pagina 'Executive Summary' con:

1. KPI: Sales Amount, Gross Margin %, Sales YoY %;
2. trend mensile Sales Amount (ultimi 24 mesi);
3. matrice Sales Amount per Region e Category;
4. barra Top 10 prodotti per Sales Amount."

Perché funziona

- Specifica chiaramente oggetti, misure, granularità, orizzonte temporale, visual desiderati
- Copilot supporta la creazione e anche la modifica di pagine (add/change/delete visuals, undo/redo)

Cosa NON aspettarsi

- Niente custom visuals e niente styling/formatting changes via Copilot
- Se il modello ha disabilitato implicit measures, Copilot non può creare report pages

Scenario 6 — DAX query view e descrizioni misure (design del modello)

Uso corretto

- Chiedere a Copilot di proporre una DAX query o spiegare una query selezionata. Copilot usa metadata del semantic model e la query selezionata
- Generare descrizioni misure: Copilot usa formula DAX e nome tabella, ma non legge commenti DAX o stringhe in doppie virgolette nella formula

5. Esempi di utilizzo errato o fuorviante

Caso 1 — Domanda ambigua: “Mostrami le vendite”

Prompt
“Mostrami le vendite per mese.”

Problema
Se nel modello esistono misure come “Sales Amount”, “Gross Profit”, “Total GPM”, “Net Revenue”, ecc., “vendite” può essere interpretato in più modi. La documentazione esplicita che in presenza di ambiguità Copilot può scegliere una metrica diversa da quella “standard” del team (es. GPM come interpretazione di sales).

Correzione

- Rendere esplicito nel prompt: “Sales Amount”
- Ridurre lo spazio d’ambiguità con **AI data schema** (escludere misure “disturbanti”) e/o **AI instructions** (“Sales = NetAmount”)

Caso 2 — Aspettativa errata: chiedere “perché” o “prevedi”

Prompt
“Perché le vendite scendono ogni luglio?”
“Quante unità venderemo il prossimo anno?”

Problema
Queste richieste implicano anomaly detection e forecasting, che la capability “Ask Copilot for data from your model” non supporta oggi.

Correzione

- Riformulare in modo “descriptive”: “Mostrami vendite per mese negli ultimi 3 anni e confronta luglio vs media annuale”; poi usare analisi e interpretazione umana (o altre funzionalità/servizi dedicati)

Caso 3 — Fidarsi dei filtri pagina/slicer “correnti”

Prompt
“Nel contesto attuale della pagina, mostrami Sales Amount per Category.”

Problema
Per la capability “Ask Copilot for data from your model” è indicato che non applica i filtri/slicer attualmente presenti nella pagina alle risposte generate nel pane. Questo può produrre una risposta “global” quando l’utente si aspetta “scoped”.

Correzione

- Esplicitare nel prompt il filtro: “per Region = X, Channel = Y, periodo = …”, oppure usare verified answers con filtri disponibili

Caso 4 — Domande fuori perimetro dati → risposta “da conoscenza generale”

Prompt
“Qual è la crescita attesa del mercato retail in Italia e come impatta le nostre vendite?”

Problema
Parte della domanda non è necessariamente nel semantic model; Copilot può rispondere usando la “general knowledge” del LLM se non trova grounding nel modello. Il rischio è una risposta plausibile ma non verificabile con i dati aziendali.

Correzione

- Separare le due domande: (1) analisi dati interni; (2) ricerca esterna (processo diverso, fonti citate, governance diversa)

Caso 5 — Richieste non supportate nella creazione pagine

Prompt
“Crea una pagina con custom visual X e applica uno stile specifico con font e colori corporate.”

Problema
Custom visuals e styling/formatting changes non sono supportati nella creazione/modifica pagine tramite Copilot.

6. Best practices per progettare modelli semantici “Copilot-friendly”

Di seguito vengono raccolte best practices che sono intenzionalmente concrete e applicabili; non sostituiscono le best practice tabular classiche, ma le enfatizzano dove Copilot è più sensibile.

6.1 Naming: ridurre ambiguità “linguistica” prima che tecnica

Copilot usa i nomi dei campi come indicazione forte per scegliere cosa usare nei visual/risposte.
Linee guida:

- **Misure**: nomi autoesplicativi e unità implicita nel nome (es. “Sales Amount”, “Gross Margin %”, “Orders Count”)
- **Colonne**: evitare “Value”, “Amount”, “Flag” senza contesto. Preferire “Net Amount”, “Order Status”, “Return Flag”
- **Date**: DimDate[Date] come riferimento unico; se avete più date (Order/Ship/Invoice), nominarle in modo esplicito e decidere una “primary” per analisi standard (e spiegarlo nelle AI instructions)

Cosa evitare:

- abbreviazioni interne (es. “GPM”, “NR”, “Cst”) senza sinonimi/istruzioni
- omonimie tra tabelle e colonne (“Region” in più tabelle senza gerarchia chiara)

6.2 Struttura modello: semplificare il mapping, non solo le performance

- Preferire **star schema** e relazioni 1→* ben definite
- Limitare many-to-many e bi-directional dove non necessario: sono gestibili dagli esperti, ma amplificano ambiguità per un sistema che può trovarsi a dover “indovinare” intenti

6.3 Linguistic modeling: sinonimi come “contratto semantico”

Copilot può essere guidato con sinonimi per termini business-specific, migliorando la comprensione delle domande. Come la documentazione suggerisce, l’autore può guidare Copilot con naming e sinonimi e rimanda al tooling Q&A/linguistic schema.

Se usate Copilot per generare sinonimi per Q&A, è utile sapere che (in quel contesto) vengono inviati a Copilot i metadati del modello (nomi tabelle/campi) e non i contenuti riga o le

domande degli utenti.
In ogni caso: i sinonimi vanno revisionati e approvati, non “accettati in blocco”.

6.4 Descrizioni: utili, ma sapere dove incidono

Le descrizioni di tabelle/colonne possono dare contesto, ma la documentazione ricorda che le descrizioni sono usate **solo** in DAX queries e nelle capability di Copilot search (non come panacea per tutte le chat skills)

Implica una scelta pragmatica:

- se il vostro focus è “ask data questions”, investite prima su AI data schema / instructions / verified answers
- se il vostro focus è “DAX query view” e governance del semantic layer, descrizioni sono un investimento importante

Quando Copilot:

- genera una DAX query,
- spiega una query selezionata,
- aiuta nella DAX Query View,

le descrizioni diventano rilevanti perché:

- forniscono contesto semantico per interpretare il ruolo di una tabella o di una misura;
- aiutano Copilot a spiegare “cosa fa” una misura o una parte della query in termini più leggibili.

In questo scenario, le descrizioni arricchiscono la spiegazione, ma:

- non cambiano la logica del calcolo,
- non correggono relazioni o misure sbagliate.

Sono un supporto cognitivo, non una leva di disambiguazione primaria.

Nelle esperienze di Copilot search (standalone o a livello di app), le descrizioni:

- aiutano Copilot a capire *di cosa parla* un semantic model o un report,
- migliorano il matching tra la richiesta dell’utente e i contenuti disponibili.

In questo caso, la descrizione è usata come segnale testuale, simile a:

- titolo,

- nome,
- metadata di alto livello.

6.5 “Prep data for AI”: usare gli strumenti nel modo giusto (e nell’ordine giusto)

Il set di funzionalità per “preparare i dati per AI” ha l’obiettivo di ridurre l’ambiguità semantica del modello e aumentare la probabilità che Copilot produca risposte corrette e coerenti con le definizioni di business. La documentazione sottolinea esplicitamente che queste funzionalità non rendono Copilot deterministico: l’interazione resta probabilistica e sensibile al contesto e al linguaggio naturale. Il loro ruolo non è garantire sempre lo stesso output, ma restringere lo spazio delle interpretazioni possibili e rendere i risultati più stabili, prevedibili e governabili in scenari reali.

Ordine consigliato

1. **AI data schema** (selezionare subset di tabelle/campi/misure su cui Copilot deve ragionare)
2. **Verified answers** (domande frequenti/critiche con risposte visual “human-approved”)
3. **AI instructions** (regole di interpretazione, terminologia, priorità tra tabelle)

AI data schema

- Selezionare campi “puliti” e con bassa ambiguità; rimuovere campi confondenti
- Nota importante: lo schema AI non si applica a tutte le capability; per creare pagine report, fare search, o usare DAX query, Copilot richiede l’intero semantic model e non considera l’AI data schema

AI instructions

- Usarle per definizioni (“Sales = NetAmount”), regole (“busy season”), priorità tra tabelle, e richieste di chiarimento (“se chiedono sales per prodotto, chiedi location”)
- Limiti importanti: non visibili e non disabilitabili dagli utenti finali; massimo 10.000 caratteri

Verified answers

- Definirli come “risposte curate” per domande ad alto impatto; Copilot li restituisce quando trova match con trigger phrase (anche semanticamente simile)

- Considerare limiti e supporto filtri: fino a tre filtri “available to users”, slicer non supportati, alcuni filtri data (relative date) non supportati; esistono limiti di quantità (es. 250 verified answers per model, 15 trigger per answer)

6.6 Mark “Approved for Copilot” e governare la discoverability

Dopo aver preparato e validato il semantic model, è possibile marcarlo come Approved for Copilot. Questo flag non modifica il comportamento analitico di Copilot, ma ha un ruolo di governance: segnala che il modello è una fonte affidabile e approvata per l’uso nelle esperienze AI. Nel Copilot standalone, i contenuti basati su modelli approvati non sono soggetti ai meccanismi di “friction treatment” previsti per fonti non governate, rendendo l’interazione più fluida e diretta. Poiché l’approvazione è applicata a livello di semantic model, lo stato si riflette automaticamente anche su tutti i report che lo utilizzano, garantendo coerenza nell’esperienza Copilot a prescindere dal punto di accesso.

Con friction treatment si intendono i meccanismi di attrito introdotti da Copilot per le risposte basate su contenuti non esplicitamente approvati. Questi meccanismi non modificano i calcoli né l’accesso ai dati, ma incidono sull’esperienza utente, introducendo segnali di cautela o limitazioni nell’uso delle risposte, al fine di distinguere fonti governate da fonti non validate. Marcare un semantic model come Approved for Copilot rimuove questo attrito, trattando il modello come fonte affidabile a livello enterprise.

A livello amministrativo esiste anche una tenant setting per **limitare** la standalone experience alla ricerca di contenuti “Approved for Copilot”.

Questa è una leva di governance molto concreta: evita che Copilot “peschi” da modelli non curati, sandbox o legacy con definizioni non allineate.

6.7 Governance e adozione: abilitazione “a fasi”, non “switch on”

La documentazione insiste su due aspetti:

- non basta abilitare Copilot e aspettarsi benefici automatici; servono training, enablement, monitoraggio e governance dell’uso
- abilitare per security group e rollout progressivo è una pratica raccomandabile

Un dettaglio sui può aver senso fare attenzione: **content filtering** può bloccare chiamate LLM contenenti certe parole/frasi; se queste compaiono nel vostro schema/metadati, Copilot può

andare in errore sistematicamente. Questo non è un “edge case teorico”: è esplicitato come rischio reale.

7. Strategie per chi ha modelli esistenti (anche legacy)

In contesti enterprise è comune avere semantic models “storici” con:

- naming tecnico (colonne “VAL”, “DT”, “KPI1”)
- molte misure ridondanti
- relazioni non star-schema
- regole business che risiedono “nella testa” di poche persone

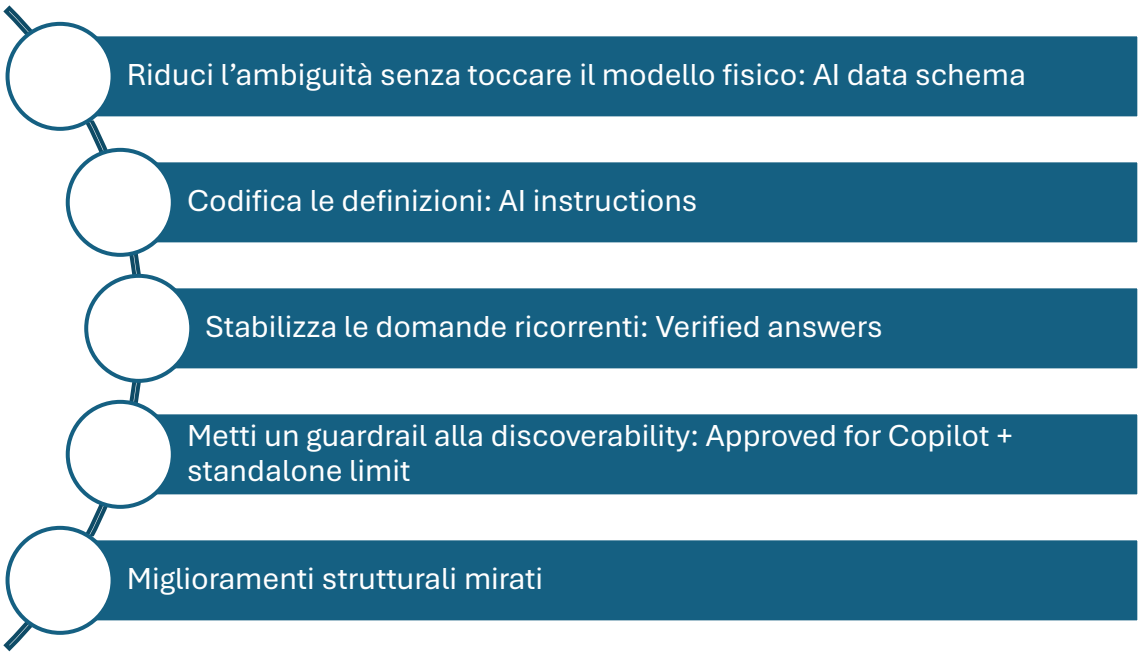
L’obiettivo qui non è “rifare tutto”, ma massimizzare il rapporto valore/sforzo.

7.1 Primo vincolo: prerequisiti di piattaforma e capacità

Prima di tutto, verificare che Copilot sia effettivamente utilizzabile nel vostro scenario:

- Copilot richiede abilitazione amministrativa in Fabric/tenant settings
- servono capacità/supporto region: requisiti generali includono capacità Fabric almeno **F2**
- trial SKUs/capacities non supportati; sovereign clouds non supportati

7.2 Strategia “incrementale” in 5 step



Step 1 — Riduci l'ambiguità senza toccare il modello fisico: AI data schema
È il quick win più efficace perché:

- non richiede riscrittura DAX
- può escludere campi/misure "pericolosi" (ambigui, duplicati, tecnici)
- migliora subito la qualità delle risposte nelle skill che usano lo schema

Step 2 — Codifica le definizioni: AI instructions
Usale come "glossario operativo" e "regole di analisi":

- mapping termini business → campi corretti
- priorità tra tabelle
- regole costanti (escludi cancellati, segno resi, valuta)

Step 3 — Stabilizza le domande ricorrenti: Verified answers
Se avete 10–20 domande che "fanno l'80% del valore" (KPI mensili, pipeline, top customer, ecc.), i verified answers:

- riducono i rischi di interpretazioni variabili
- rendono l'esperienza ripetibile e "trusted"

Step 4 — Metti un guardrail alla discoverability: Approved for Copilot + standalone limit

- Marca "Approved for Copilot" i modelli curati
- abilita "only show approved items" nel standalone se vuoi evitare che Copilot peschi da modelli legacy non curati

Step 5 — Miglioramenti strutturali mirati
Solo dopo i quick win sopra, intervenire dove serve davvero:

- Rinominare misure "canoniche" (non tutte): Sales Amount, Gross Margin %, Orders Count
- Introdurre una DimDate robusta se mancante (impatterà fortemente la qualità delle domande temporali)
- Creare misure YoY/YTD standard per evitare che Copilot generi continuamente calcoli ad hoc

7.3 Criteri per decidere cosa sistemare subito vs mai

Alta priorità (impatto immediato su Copilot)

- termini ambigui e duplicati (soprattutto “sales”, “margin”, “cost”)
- assenza di misure base canoniche
- assenza/caos di date
- campi tecnici esposti all’utente

Bassa priorità (se Copilot non è il focus primario)

- refactoring profondo relazioni se lo schema è già “stabile” e toccarlo è rischioso
- ottimizzazioni puramente di performance non legate alle query che Copilot genera

7.4 Nota DevOps/ALM: LSDL, Git e deployment pipelines

Le modifiche ad AI data schema e AI instructions vengono salvate a livello di semantic model, non nel singolo report. In particolare, queste configurazioni diventano parte della definizione strutturale del modello (persistite nel formato interno LSDL – Local Semantic Definition Language), seguendo il modello in tutti i report che lo utilizzano e nel suo intero ciclo di vita.

8. Approccio mentale corretto all'uso di Copilot

8.1 Copilot cambia l'interfaccia, non la responsabilità

Copilot rende più facile "chiedere" e "prototipare", ma non elimina:

- responsabilità di definire metriche e semantica
- validazione dei risultati e controllo del contesto (filtri, granularità)
- governance (quali modelli sono affidabili, quali sono certificati/approved)
- alfabetizzazione dei decision maker (sapere leggere e interpretare un visual)

La documentazione è esplicita sul fatto che Copilot **non sostituisce** chi crea semantic model/report e che serve un rollout governato con training e valutazione output.

8.2 Copilot come "amplificatore": workflow consigliato

Un workflow pratico:

1. **Chiedi** con prompt specifici (misura, dimensione, periodo, filtri)
2. **Ispeziona** "How Copilot arrived at this" e/o DAX query generata
3. **Confronta** con visual standard o misure canoniche (se esistono)
4. **Industrializza**: se la domanda è ricorrente, trasformala in una misura/pagina/verified answer (non lasciarla in chat)
5. **Governa**: "Approved for Copilot", limitazione standalone, monitoraggio capacity/consumi

8.3 Prompting: meno "prompt engineering", più "semantic engineering"

Il valore non sta nel trovare la frase perfetta; sta nel progettare un modello dove:

- i concetti business sono univoci
- le misure "canoniche" esistono e hanno definizioni stabili
- Copilot ha istruzioni/schema/verified answers per non dover "indovinare"

9. Curiosità, dettagli poco noti e consigli pratici

Questa sezione raccoglie aspetti che spesso emergono solo nell'uso quotidiano e che aiutano a evitare falsi positivi/negativi.

9.1 "Clear chat" non significa "risposta diversa"

- "Clear chat" rimuove il contesto conversazionale, utile quando cambi argomento
- In alcuni casi, se viene posto lo stesso prompt (o uno semanticamente equivalente) su un semantic model invariato entro una finestra temporale limitata, Copilot può riutilizzare una risposta precedente invece di rigenerarla. L'operazione di clear chat azzera il contesto conversazionale, ma non garantisce l'invalidazione di eventuali meccanismi di caching a livello di servizio. Per ottenere un risultato diverso è quindi necessario modificare in modo significativo il prompt o intervenire sul modello o sui dati sottostanti

9.2 "Prep data for AI" vive sul semantic model, non sul report

- Tutti gli update del tooling sono salvati sul semantic model
- Le modifiche possono richiedere tempo per propagarsi; in Desktop è consigliato chiudere e riaprire il Copilot pane per vedere l'effetto, e in alcuni casi si parla di minuti, fino a finestre più ampie in scenari complessi

9.3 AI data schema: non è "universale"

L'AI data schema:

- è invisibile ai consumer
- si applica solo alle capability che lo usano
- non viene considerato in creazione pagine, search, o DAX query (dove serve l'intero modello)

Attenzione che se il problema è "Copilot sbaglia campo quando crea una pagina report", l'AI data schema potrebbe non essere la leva corretta; in quel caso entrano in gioco naming, relazioni, misure e (in parte) instructions.

9.5 Verified answers: attenzione a filtri e manutenzione

- I verified answers salvano filtri persistenti applicati al visual in fase di setup; slicer non supportati
- Alcuni filtri data (relative date) non sono supportati

- Sono model-level: cambiare il visual nel report non cambia automaticamente il verified answer; per cambiare, si gestisce dal dialog di management (o si ricrea)

9.6 “Ask data questions” (model-based)

Per la capability “Ask Copilot for data from your model”:

- è (ad oggi) disponibile nel service; supporto Desktop “coming soon”
- lingua supportata: English
- non applica filtri/slicer correnti del report alle risposte nel pane
- non supporta richieste tipo forecasting/anomaly detection

Questi dettagli devono entrare nella comunicazione agli utenti, altrimenti Copilot viene giudicato “inaffidabile” per motivi di perimetro, non di qualità del modello.

9.7 Creazione/modifica pagine: prerequisiti e blocchi

- Q&A switch deve essere ON per report creation/editing
- non funziona con real-time streaming, live connection ad Analysis Services, implicit measures disabled
- niente custom visuals e niente styling via Copilot

Questo è un punto di architettura: se la vostra governance impone implicit measures disabled per qualità, accettate che Copilot page generation non sia una capability per quel modello (o valutate modelli “authoring-friendly” separati).

9.8 Standalone Copilot: autoselezione workspace e implicazioni capacity

La documentazione descrive che, in assenza di Fabric Copilot capacity (FCC), lo standalone può selezionare automaticamente un workspace idoneo per usage tracking/billing, con una logica pesata sulla capacità disponibile; l’utente può cambiarlo.
Per architetture enterprise: questo è un motivo forte per valutare un **FCC dedicato** quando l’uso cresce.

9.9 Operatività: tempi di riconoscimento capacità e rollout

- Nuove capacità o scale-up possono richiedere fino a 24 ore perché Copilot risulti disponibile/riconosciuto
- Anche alcune feature possono essere in rollout progressivo (es. nel 2025 la preparazione dati per AI è stata estesa al service con rollout globale)

9.10 Direzione di marcia: Q&A “classico” deprecato, Copilot come alternativa

Le esperienze Q&A sono indicate come in dismissione a dicembre 2026, con raccomandazione di migrare a Copilot per query in linguaggio naturale.
Questo sottolinea quanto stiamo scrivendo: investire oggi nella qualità semantica (schema, sinonimi, AI tooling) non è un “optional”, quanto una preparazione strutturale a come gli utenti interagiranno con i dati.

Da dove iniziare

Se Copilot è già abilitato (o sta per esserlo), il primo passo non è “fare prompt migliori”, ma guardare il semantic model con occhi diversi. Domani mattina, scegli un modello realmente usato in produzione e chiediti: esistono misure chiaramente canoniche per i KPI principali? I nomi dei campi raccontano davvero il significato di business, o richiedono spiegazioni verbali? Se emergono ambiguità, quello è il punto di partenza.

Il secondo passo è rendere esplicito ciò che oggi è implicito. Usa gli strumenti di *Prepare data for AI* non per “istruire l’AI”, ma per codificare decisioni già prese: cosa intendiamo per vendite, quale data è quella di riferimento, quali filtri sono sempre validi. Anche poche AI instructions ben scritte e un AI data schema minimale possono fare una differenza immediata nella qualità delle risposte.

Il terzo passo è osservare Copilot come strumento di test, non come oracolo. Usalo per porre domande che normalmente generano discussione tra analisti e stakeholder. Se Copilot risponde in modo inatteso, non è un fallimento: è un segnale. Sta mostrando un’ambiguità del modello che vale la pena correggere prima che produca effetti a valle.

Infine, investi tempo nella formazione mirata, non generica. Insegna agli utenti chiave come porre le domande, come interpretare le risposte, come verificarle, e quando fidarsi. Insegna a chi costruisce i modelli semantici e a chi sviluppa i report come progettare pensando anche a Copilot. È questo passaggio che trasforma Copilot da funzionalità abilitata a capacità realmente adottata.

Iniziare da qui non richiede una rivoluzione, ma un cambio di prospettiva: trattare Copilot non come un esperimento isolato, bensì come uno specchio della maturità del proprio ecosistema dati. Ed è proprio da questo primo, piccolo passo che nasce il valore più duraturo.

Conclusioni

Copilot in Power BI rappresenta un passaggio importante nel modo in cui le organizzazioni possono accedere, esplorare e comprendere i propri dati. Non perché sostituisca il lavoro umano, ma perché rende più rapido e diffuso l'accesso a capacità analitiche che in passato richiedevano competenze specialistiche e tempi più lunghi. Se inserito in un contesto solido, Copilot può diventare un acceleratore reale di insight, collaborazione e qualità delle decisioni.

Il valore di Copilot, però, non risiede nel prompt o nell'interfaccia, ma nella qualità del contesto su cui opera. Un semantic model chiaro, governato e coerente trasforma Copilot in uno strumento affidabile; un modello ambiguo ne amplifica invece le fragilità. In questo senso, Copilot non è solo una nuova funzionalità, ma anche un potente catalizzatore di maturità: rende evidenti le scelte architetturali, le definizioni di business e le pratiche di governance che spesso restavano implicite.

Copilot è un moltiplicatore della qualità del vostro modello semantico, non è un sostituto.

L'adozione consapevole di Copilot non richiede una riscrittura radicale dei sistemi esistenti, ma un percorso progressivo: chiarire le metriche fondamentali, ridurre l'ambiguità semantica, investire in formazione e validazione, e utilizzare gli strumenti di governance messi a disposizione dalla piattaforma. È un investimento che paga nel tempo, perché migliora non solo l'esperienza con Copilot, ma la qualità complessiva dell'ecosistema dati.

Guardato in questa prospettiva, Copilot non è una scorciatoia, ma un moltiplicatore. Non elimina la necessità di comprendere i dati, ma rende quella comprensione più accessibile, condivisa e verificabile. Ed è proprio in questo equilibrio tra automazione e responsabilità che risiede il suo valore più duraturo.