Snowflake Semantic Views Best Practice Guide

Thisguide providespracticalimplementationguidanceforSnowflake Semantic Views at Canva, for deployment into the Canva-wide Semantic Layer.

IMPORTANT: Semantic Views may appear simple, with a wizard-like GUI, but if you create Semantic Views “on autopilot” you may end up with poor semantic views.
It is important to take the time to read these best practices and apply them when implementing the Semantic Layer

Be Cautious with the Snowflake GUI Wizard

The Snowflake Semantic View GUI editor/wizard can be a useful starting point, but it frequently creates poor semantic views by default:

- AI-generated metadata is unreliable - The wizard offers to use AI to generate metadata such as synonyms but does so extremely poorly. These auto-generated synonyms are often inaccurate or misleading and will hurt your semantic view's per for mance.
- Defaults to Facts, not Metrics - The wizard tends to create Facts for all numeric columns, despite Snowflake's own documentation describing Metrics as the primary way users interact with Semantic Views. You'll need to manually convert these to proper Metrics.

Recommended approach: Use the GUI wizard to generate initial YAML structure, then immediately:

1. Delete all auto-generated synonyms and sample values
2. Convert Facts to Metrics with proper aggregation logic
3. Manually add meaningful descriptions, synonyms, and metadata following the guidelines in this document

The Semantic View Editor GUI is fine for viewing a Semantic View or getting started - but do not trust its defaults

Table of Contents

Key Documents

- Semantic Layer - Data Architecture
- Semantic / AI Readiness Framework for Data Models
- Semantic Layer Roll-Out
- Semantic Layer Governance

General Principles

Principle	DO	DON’T
Intentional Every semantic model must satisfy a business use case.	Start with 10+ validation questions in mind before writing any YAML	"Let's expose all our tables and see what happens"

Consistent A single source of truth in the semantic layer for any problem domain, even if it means refactoring existing DWH objects to achieve this.	One definition per metric	Many competing ARR calculations
Empowering expertise The semantic layer for any business domain is owned by the respective embedded data team.	E.g. Marketing AEs/DSes owns marketing semantic views	Global semantic view with central team trying to model all domains
Curated and Understandable Every semantic model must be readable, reviewed and approved by a human	Ensure a human at least reviews the semantic view before it reaches production	Use AI to generate the semantic view without thorough human review
AI-ready by design Optimised for LLM consumption	Write descriptions like "explaining to an intern who just started"	Assume the AI will "figure it out"
Verified Tests/evaluations monitored by owning team	Automated validation question testing in CI/CD	"It worked once in dev"
Relevant Usage monitoring and lifecycle planning	Perform regular usage reviews with sunset procedures	Zombie views that nobody uses, confusing landscape of data, lack of clarity whats' supported and what isn't
Performant Optimized for query performance and cost	Provide a good user experience - model data for performant access	"Why does this simple question take 5 minutes?"
Discoverable Easily findable through clear documentation, naming conventions, metadata, and lineage.	Follow naming conventions, add comprehensive metadata, maintain documentation	"Ask Sarah, she knows where everything is"
Quality over Quantity Include only the minimum required - this goes for components of the semantic view as well as custom instructions and other context.	Only add what is required and what produces demonstrable improvement	Pre-emptively generating an ultra-verbose custom instructions and SV descriptions "just in case"

Planning Your Semantic View

Start with Questions, Not Tables

IMPORTANT: Do not plan to rush out and expose all of your existing data via the Semantic Layer

While the goal is to eventually have a Canva-wide semantic layer, rich with both breadth and depth of data - everything included should be:

- carefully considered
- part of a cohesive "semantic model"
- Based on real reporting need, avoid data dumps "just in case"

Suggested starting point:

1. Collect 10+ real business questions from stakeholders
2. Identify required metrics and dimensions
3. Map to minimum necessary tables and columns

Data Modelling for the Semantic Layer

The Semantic Layer works best with proper dimensional modelling i.e. tables conforming to a star schema. For more information see: Dimensional Modeling

Ensure your data models are well-engineered and and in a good state before registering in the Semantic Layer.

See: Semantic / AI Readiness Framework for Data Models

Model evaluation tool: aim to score a minimum of 7/10 using Semantic Layer - Data Model Readiness Evaluation Prompt

Domain Scoping

We are organising our Semantic Layer into “Subject Areas” based on the identified Data Domains across Canva. This aligns with the our Warehouse Data Architecture which uses Data Domains for organising Data Services.

In general, the Semantic Layer will consist of one or more semantic views per Subject Area (See:Semantic Layer - Data Architecture).

This will require data teams establishing clear boundaries to make ownership clear and prevent overlap of metrics across different Subject Areas.

Guidelines to Prevent Overlapping Metrics / Dimensions

Metrics

- Metric Ownership - Before surfacing a metric in the Semantic Layer, ask yourself - does my team/group/Supergroup own this metric? If not, consider why you are exposing this in your Subject Area - should you instead be a consumer of another domain’s Semantic Views and if you have specific required e.g. segmentation, would they be willing to incorporate this into their Subject Area. If you genuinely need to surface a metric that has overlap with a metric in another subject area, make the difference strongly explicit in your metric - giving it a distinct name and describe the purpose and any differences clearly in metric metadata.
- Report Models - be deliberate when exposing report models (i.e. flat, non-dimensional tables). If a true star schema performs poorly, don’t assume you need to create a report model - first consider using summary fact tables that are still dimensional in style but aggregate data. Reserve report models only for cases where dimension joins cause performance issues.
- Unique Metric Names - CI will block the same metric being defined multiple times in a single Semantic Layer Semantic View, and this also applies to synonyms.

Dimensions

- Regular Dimensions vs Degenerate Dimensions - For dimensions, always favour discrete, stand-along dimension tables over degenerate dimensions
- Degenerate Dimensions - Take care with degenerate dimensions - if the degenerate dimension is used in multiple fact tables, do not change the values in the field - if you do, then name the column explicitly to differentiate it from the base field.

Related content

- Required dimensions for Metrics Platform
- Canva’s Warehouse Data Modelling Standards

Building Semantic Views

Component Implementation Guide

Component	Guidelines	Naming Convention	Example
Logical Tables	Eligible Tables/Views • Only reference tables from public production schemas: model, report, dm_* (CI check) • Check your model against the Semantic / AI Readiness Framework for Data Models to ensure it is suited for inclusion in the Semantic Layer	• Mirror the name of the underlying warehouse model (fact_, dim_, report_) • Name must be unique within the semantic view • Keep under 50 characters • Use singular form (fact_subscription_event not fact_subscription_events)	

	• Include only what's necessary to answer your target questions - resist the urge to add "everything" • Follow star schema structure: facts at center, dimensions around edges Role-playing dimensions • A single physical dimension table should be included multiple times as seperate logical tables for different business purposes. Do not create multiple relationships/join paths between a fact and dimension (Snowflake allows this on Semantic View creation, but it will fail when queried). dbt Materialisation • Must use `{{ ref() }}` or `{{ source() }}` - no hardcoded table names Table Descriptions • Each table needs a clear description explaining its grain and purpose	• For role-playing dimensions: Give each logical role a distinct, business-meaningful name ◦ Physical table: `dim_date` ◦ In semantic view: create multiple logical tables for `dim_order_date`, `dim_ship_date`, `dim_delivery_date` etc.	
Relationships	• Only between fact and dimension tables - no dim↔dim, fact↔fact, or report↔report relationships • Joins should be based on a single `dkey` column • Every dimension table must have at least one relationship with a fact • Snowflaked dimensions are not allowed (multi-hop joins) • Joins are where LLMs struggle most - show examples in verified queries • One relationship per fact-dimension pair (fact being the right table and dim being the left table)	<fact table name>__<dim_table_name>	
Dimensions	MUST include sample values OR link to Cortex Search Service (CI check) - can't have neither No dkey columns exposed as dimensions (CI check) - dkeys are for joins only, not analysis Dimension Descriptions • Explain what the attribute represents and its business meaning • Be careful with embedded examples - sample values in dimension descriptions can override	• Names must be unique within the semantic view (Exception: time dimensions, degenerate dimensions across facts) • In general, align to the names in underlying tables, but note best practices: ◦ Use business-friendly names, not technical field names ◦ Keep under 50 characters (CI check) ◦ No generic names like `type`, `date`, `id`, `day`, `_id`	

	Cortex Search Services or cause incorrect routing • Avoid terminology that overlaps with other component names (e.g. don't say "geographic region" in a market description if you have a separate region dimension) Sample Values • Include either sample values or a Cortex Search service - not both • If the sample values contain the complete set of possible values, set `is_enum` to indicate this (see: docs) • Avoid setting sample values unnecessarily e.g. TRUE/FALSE for booleans or arbitrary numbers such as 1212.4 for numeric fields.	◦ Be specific: "subscription plan" not just "plan"	
Time Dimensions	• In general, every fact should have a time dimension defined or link to a dimension that has a time dimension. • For Accumulating Snapshot Fact tables, clearly indicate which time dimension relates to which metrics. TODO: verify how much is necessary (metric description / custom instructions etc) • A common pattern when dim_date is linked to a fact table: create a Time Dimension on dim_date.calendar date	• Be specific: `subscription_start_date`, `event_date`, `snapshot_date` - avoid just `date` • Unless it conflicts with the above, match the naming from your underlying table for clarity	
Metrics	• Metrics are the core of your semantic view - invest time to get them right - in particular additivity (see section below) • Only define on fact tables or report models, NEVER on dimensions • Every fact table MUST have at least one metric • Give metrics unique, descriptive, non-generic names - avoid names such as “user count” or “amount”. Metric Descriptions • Explain what's being measured AND how it's calculated	• Business-friendly names that are self-explanatory • Names must be unique within semantic view - and if possible, unique across Canva • Be extremely explicit in the metric naming to avoid collisions with existing metrics - “Monthly Active Video Users”, “B2B Annualised Recurring Revenue” • Keep under 50 characters • Use action-oriented language where relevant: "new subscriptions", "active users", "revenue"	
Named Filters	• Use for commonly applied business rules or segments where the filtering on an existing dimension is not straight forward.	• Names must be unique within semantic view • Descriptive names that explain the filter condition	

	◦ eg: a specific complex filter of multiple fields or a specific value from a field where the value is not as obvious as it's business meaning. • Snowflake works best with structured, explicit filters rather than relying on custom instructions to interpret intent • Test that filters work as expected in verified queries Filter Descriptions • Give filters clear, descriptive names that help the LLM understand when to apply them		
Facts	Avoid using Facts and always prefer Metrics		

Cortex Search Services	• Use for dimensions with high cardinality or where fuzzy matching is needed (e.g. company names, product names, campaign names) - significantly improves accuracy for natural language queries • Required if dimension doesn't have sample values • Must be deployed separately BEFORE referencing in your semantic view • Link the search service to specific dimensions in your semantic view configuration. Note the index does not need to be on the field e.g. a cortex search on a small table can be configured for a degenerate dimension on a fact table with high data volumes (assuming the values are in sync e.g. user_journey) • Test that the search service is being used correctly - check generated SQL to verify using Cortex Analyst Playground • Less useful for: low-cardinality dimensions where sample values work fine (e.g. subscription tiers, countries) • Implement a Cortex Search Service per attribute, using the ON clause • Cost consideration: Cortex Search Services require regular syncs, so use intentionally where the accuracy benefit justifies it	`ctx_search__<table_name>__<search column_name>`	`foundation.semantic.ctx_search__ref_currency__currency_code`

Key Attributes

These attributes are common across many components - the following are general guidelines (Component-specific details relating to these fields are in the table above)

Attribute	Guidelines	Naming Convention	Example
Description	• Every component must have a description • Be direct, do not start with “This table…” or “This metric….” • Write factually, with brevity without sacrificing clarity - consuming tools may display the description in pop-ups or UI with limited space. • Write as if explaining to a new team member or intern who just started - this is the right level for LLMs	N/A	

	• Be specific and unambiguous - vague descriptions lead to misinterpretation • Include what the thing IS and what it's used for		
Synonym	• Always provide synonyms • Synonyms help the LLM understand alternative ways users might refer to a concept • Must be unique across the entire semantic view • Include common business terms, abbreviations, and natural language variations • Think about how different teams or stakeholders might refer to the same thing • Avoid overly technical terms unless your audience uses them • Don't go overboard - quality over quantity (3-5 synonyms typically sufficient) • Test that synonyms route correctly by including them in your validation questions	• Use spaces, NOT underscores (e.g. "monthly active users" not "monthly_active_users") • No quotation marks	

Handling Non-Additive and Semi-Additive Metrics

This section addresses the complexity of metrics that cannot be simply summed across all dimensions.

Understanding Additivity

Type	Description	Examples	Snowflake Approach
Additive	Can be summed across all dimensions	Revenue, count of orders	Standard SUM aggregation
Semi-Additive	Can be summed across some dimensions but not others (typically not time)	ARR, MAU, account balance	NOT CURRENTLY SUPPORTED (Feature expected during H2 2025) Workaround: grain-specific metrics
Non-Additive	Cannot be meaningfully summed	Conversion rates, averages, percentages	Calculate from components in Semantic View Ensure aggregation via Semantic Query

See Composability vs Additivity for metrics and aggregations for further detail

Non-Additive Measures

- Ensure you implement the logic correctly as a single Metric expression, to guide Cortex Analyst to in aggregating the metric correctly.
- In the case of ratios/conversion metrics, do not simply surface the components (i.e. the numerator and denominator) as separate metrics and expect downstream tools such as Cortex Analyst or BI tools to them perform the final calculation.

Semi-Additive Measures

PENDING IMPLEMENTATION BY SNOWFLAKE DUE NOVEMBER 2025

Time Intelligence Patterns for Metrics

For derived, time-based metric variants such as Year to Date (YTD), Same Period Last Year (SPLY), Month on Month (MoM), Rolling windows a Semantic Layer is the ideal location. However, Snowflake Semantic Views currently don't have specific functionality to support these kinds of metrics easily.

Example implementation of Last Month / Last Year values: <link to examples>

Semantic Context Hints

Verified Queries

The purpose of Verified Queries is to "Teach by example".

Cortext Analyst will route questions directly to a verified query if it has determined that it will answer the user's question.

Do not rely on verified queries alone - your semantic view should work well without verified queries. If you metrics only work when a verified query is used, then that is a red flag that the semantic view is not configured well. It is recommended to test your models with and without verified queries enabled.

When creating a ground truth evaluation question bank, try to have a split of questions that are answered with and some without verified queries, to reduce the risk of falsely evaluating the semantic views.

Follow Semantic Layer Observability Best Practices for guidelines on created verification questions.

Custom Instructions

For any additional context not captured in any other way (Semantic Views components, verified queries or other metadata) that can help guide AI tools to access data, use Custom Instructions.

Avoid using custom instructions for listing synonyms, sample values or general field descriptions, focus on the less-routine specifics for your data models.
Do not preemptively write lots of content that you think will help - try to get the model to work well using the native Semantic View components, and only use Custom Instructions to address issues you find.

Cortex Search Service

Cortex Search Service provides a way to improve literal string searches to help Cortex Analyst generate more accurate SQL queries. See Improve literal search to enhance Cortex Analyst responses | Snowflake Documentation

Cortex Search Service Configuration for Semantic Views

Macros have been created to enable creation of Cortex Search services. As an example, see the Cortex Search services created in the foundation project here.

For an overview of Cortex Search services see Cortext Search for Cortext Analyst Semantic Views.

TODO: CONFIRM USE OF ON vs ATTRIBUTES

Other Considerations

Surfacing your Metrics in the Metrics Platform

Initially, Semantic Views for Metrics Platform are hosted in the Metrics Platform namespace. In the near future, Metrics Platform will utilise the definitions in the Semantic Layer.

Details will be provided in future as to how to mark a metric in the Semantic Layer for inclusion in the Metrics Platform.

Materialisation Strategy: Database vs Semantic Layer

For now, Semantic Views do not have any special caching / pre-aggregation features. Therefore poor performance when querying may require developing additional, more aggregated fact tables or report models - but extreme care should be taken with this to avoid it introducing near duplicate metrics in the semantic view (i.e. the same metric at different grains/levels of aggregation).

Do not rush to aggregate all data however, first check the cardinality of your dimensions joined from any fact tables - for example, while `dim_user` is marked as a dimension, it contains billions of rows - so if you need say `user_journey` from this field (with 8 distinct values) consider implementing this as a degenerate dimension rather than leaving the join to dim_user, suffering poor performance and then assuming you need to create summary facts or report models.

See Metrics Platform - Semantic Layer Architecture and Canva's Warehouse Data Modelling Standards.

Testing & Validation

Ground Truth Framework

Building Your Test Suite

Testing Semantic Views for performance when asking questions using Cortex Analyst is critical else your work defining the semantic view is shooting in the dark. Define questions and test them to determine how well your semantic views perform.

Follow Semantic Layer Observability Best Practices for guidelines on what your validation questions should cover.

Testing Process

See: Setting up AI observability for Semantic Layer in Snowflake

Debugging Decision Tree

```
AI giving wrong answer?
└── Check generated SQL
    ├── Wrong tables? → Review relationships, logical table descriptions and custom instructions
    ├── Wrong aggregation? → Check if multiple join paths exist; review custom instructions
    ├── Wrong time grain? → Check base_table logic; if underlying data defines a different date for each measure, consider
    transitioning to a periodic snapshot fact table
    └── Missing context?
        └── Add verified query examples, custom instructions and check for coverage of sample values/cortex search services, review logical
    table and field descriptions.
```

Example Reference Semantic Views

See Semantic Layer WG - Reference Semantic Views

Related Guides

- Snowflake Semantic Views: Quick Start Guide (PIE)
- How to Develop Semantic Views for AI (Marketing Data)
- Metrics Platform - Semantic Layer Architecture
- Composability vs Additivity Deep Dive

Frequently Asked Questions

Question	Answer
How many tables can I include in a Semantic View?	There is no hard limit, and Snowflake have indicated they aim to continue to increase any underlying "token limits" for Semantic Views, with an ultimate goal to remove these limits. Therefore, size should not be a consideration, once the limit is reached, you can expand into other "overflow" Semantic Views to extend your Semantic Layer Subject Area.
Should I use report tables or fact tables with a star schema?	By default, you should always try to use a star schema, as this is the best way to ensure consistent definitions of dimension values. Report tables (pre-joined and pre-aggregated) should only be used for performance reasons when both fact and summary fact tables do not return data fast enough in response to common queries.
What should I do if I am restricted by Snowflake Semantic Views - Known Limitations ?	In general, if you need to use a feature that is confirmed to be coming soon, it is far better to wait for official support instead of trying to implement work-arounds via a multitude of "spray and pray" overly verbose custom instructions spread over the model and metric definitions or by relying solely on verified queries as these may work inconsistently. If you find a limitation that is not in the list of known limitations and/or has no timeline given by Snowflake, raise this with the Semantic Layer Working Group so we can channel this to Snowflake who we are working closely with to improve Semantic View product features.