4 main strategies

- 1. **Ephemeral** fancy way of saying "reusable SQL snippet"
- 2. View saved SQL statement in your warehouse
- 3. Table saved data table in your warehouse (my default)
- **4. Incremental** when tables are too slow to drop & replace constantly

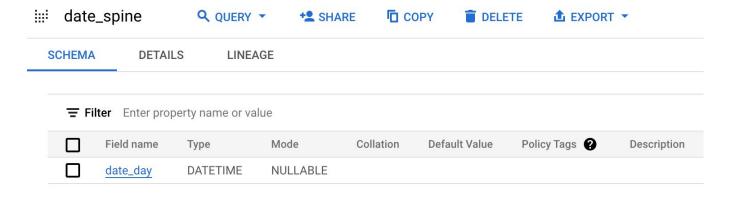
Ephemeral

For quick running SQL that's not worth materialising

```
{{ ref('int_ecommerce__first_order_created') }}
WITH
 __dbt__cte__int_ecommerce__first_order_created as (
SELECT
   user_id,
   MIN(created_at) AS first_order_created_at
FROM `ferrous-iridium-379614`.`dbt test`.`stg ecommerce orders
GROUP BY 1
```

View

Doesn't store data. Faster to run, longer to query



Table

Stores data. Longer to run, faster to query

=	order	S Q QUERY	▼ + SHAR	Е СОРУ
	SCHEMA	DETAILS	PREVIEW	LINEAGE
	〒 Filter Enter property name or value			
		Field name	Туре	Mode
		order_id	INTEGER	NULLABLE
		order_created_at	TIMESTAMP	NULLABLE
		order_shipped_at	TIMESTAMP	NULLABLE
		order_delivered_at	TIMESTAMP	NULLABLE
		order_returned_at	TIMESTAMP	NULLABLE
		order_status	STRING	NULLABLE
		num_items_ordered	INTEGER	NULLABLE
		total_sale_price	FLOAT	NULLABLE
		total_product_cost	FLOAT	NULLABLE
		total_profit	FLOAT	NULLABLE
		total_discount	FLOAT	NULLABLE

Incremental

Table appended/merged with new data, rather than drop & replace

```
{{
    config(
        materialized='incremental',
        unique_key='event_id',
        on_schema_change='sync_all_columns',
    )
}}
WITH source AS (
    SELECT *

FROM {{ source('thelook_ecommerce', 'events') }}
)
```

```
{# Only runs this filter on an incremental run #}
{% if is_incremental() %}

{# The {{ this }} macro is essentially a {{ ref('') }} macro that allows for a circular reference #}
WHERE created_at > (SELECT MAX(created_at) FROM {{ this }})

{% endif %}
```