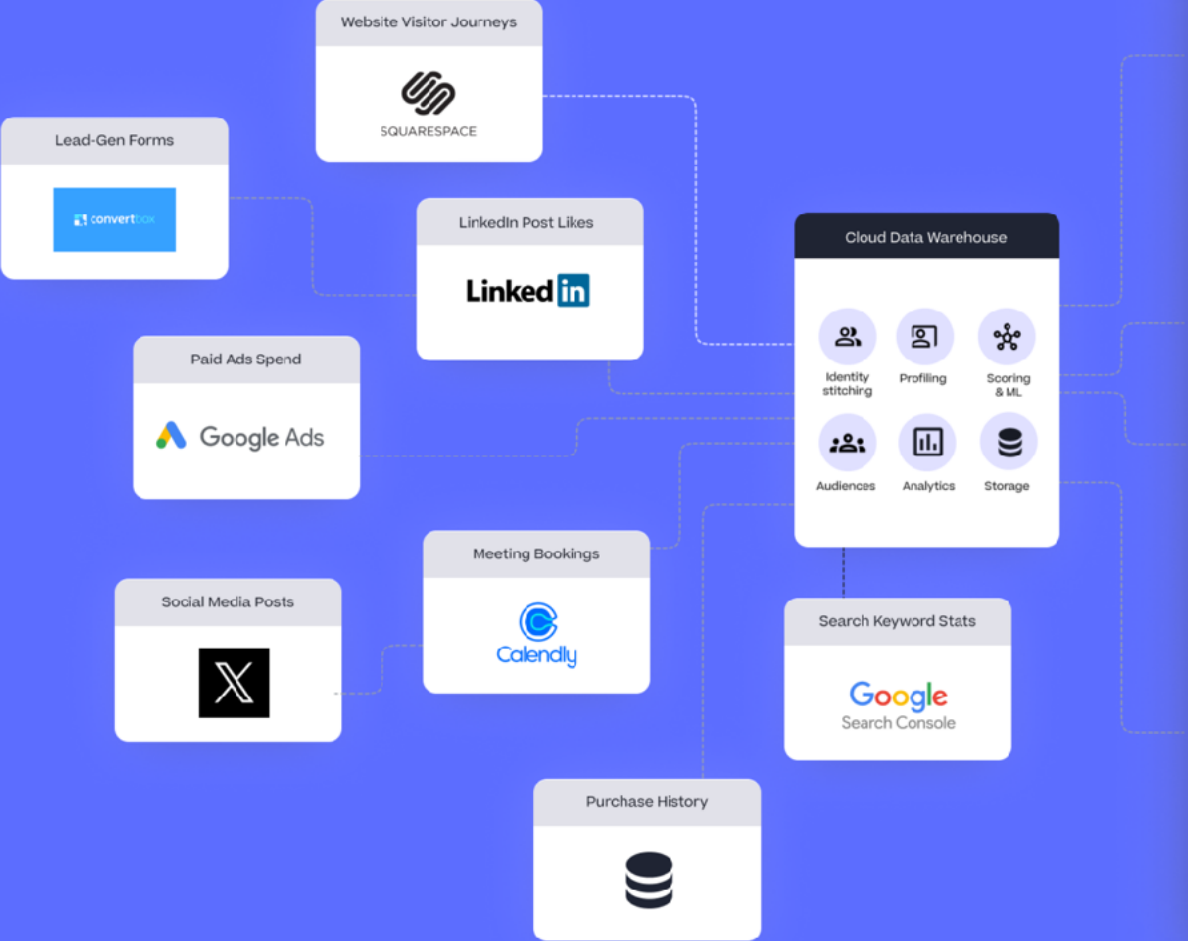


# BEST OF THE RITTMAN ANALYTICS BLOG 2023



Articles authored by:

Mark Rittman, Chief Executive Officer

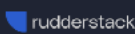
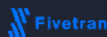
Olivier Dupius, Principal Consultant

Amir Jaber, Senior Analytics Engineer

Jordan Ilyat, Analytics Engineer



Google Cloud  
Partner



Rittman Analytics is a Google Cloud Partner that specialises in Data Analytics, Data Management and Artificial Intelligence

Our services transform customer experiences, increase efficiency and create actionable insights, making your data analytics vision a reality.

# About Rittman Analytics

We scale and upgrade your organisation's data capability

Rittman Analytics is a boutique data analytics consultancy that works with growth-stage, mid-market and enterprise businesses just like yours.

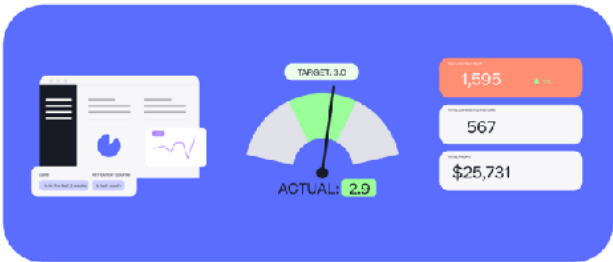
We work with you and your team to upgrade your analytics data stack, build and enable your data team and remove the reporting bottlenecks that frustrate your business users.



And once your users' basic reporting needs are met ... we'll help you harness the opportunities and innovation that Gen AI and ML now offer.

## Rapid Time-to-Value and Best-Practice Metrics

Our analytics consulting services can help your business better understand the dynamics of your CAC and LTV, identify the most cost-effective marketing channels and adjust strategies to improve the ratio for better profitability.



We can help you and your team measure and analyze the performance of your customer acquisition and digital marketing campaigns, determine the return on investment (ROI) and provide insights on how to allocate resources more effectively for future campaigns.

## Experts in Building Data Stacks for Growth-Stage Businesses

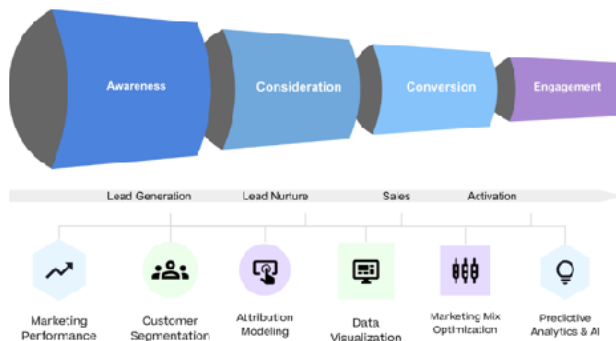
Self-service BI tools and business user-friendly semantic models used to be only available to the largest enterprises and later-stage startups, but now there are alternatives that have more features, are just as integrated and can be more affordable.

We're authorised delivery partners for Cube, Dagster, Preset, dbt Labs, Fivetran, Rudderstack and Snowflake along with Google Cloud (for Looker and Looker Studio), Oracle, Segment and Lightdash.



## Analytics for the Full Product and Customer Lifecycle

Our services cover all stages in the customer acquisition funnel from social media and content marketing analytics, through conversion optimization and marketing attribution to customer segmentation and predictive analytics.



Discover How customers use your product and what does their buying journey look like? Understand why they behave the way they do, what keeps them engaged or what's pushing them out?

Interested? - Find Out More!

Contact us at [info@rittmananalytics.com](mailto:info@rittmananalytics.com) or check-out our website at <https://rittmananalytics.com> to get in-contact - we'd love to tell you more about our services and how we can help your team scale its analytics capabilities!



# Table of Contents

Customer First-Order Segmentation using Looker and Google BigQuery	6
Building up a Semantic Layer with dbt Metrics, Cube and Droughty	12
ChatGPT, Large Language Models and the Future of dbt and Analytics Consulting	21
Building a Mobile-Friendly KPI Dashboard using Looker Studio Integration with Looker Universal Semantic Model	33
Optimizing Data Materialization Using Dagster's Policies	40
Building your Own GA4 Rules-Based Marketing Attribution Models using Google BigQuery and Looker	49
ChatGPT-Powered Data Analysis using Cube, Delphi and the Code Interpreter Plugin	54
Google Cloud Cortex Framework brings Packaged Analytics to the Modern Data Stack	76
Bringing dbt and Analytics Engineering to Oracle Autonomous Data Warehouse	86
Unlocking Data Orchestration: How to Orchestrate Fivetran and dbt with Dagster	98
Coalesce 2023 and the new dbt Cloud CLI	109
How Rittman Analytics Builds Data Stacks for Growth-Stage Businesses using Cube, Dagster and Preset	112
How Rittman Analytics Does Web Analytics and Marketing Attribution, using Google BigQuery, Looker, dbt and Segment	120
10 Ways Your Modern Data Stack Project Can Fail	131

# Customer First-Order Segmentation using Looker and Google BigQuery

Mark Rittman, January 2023

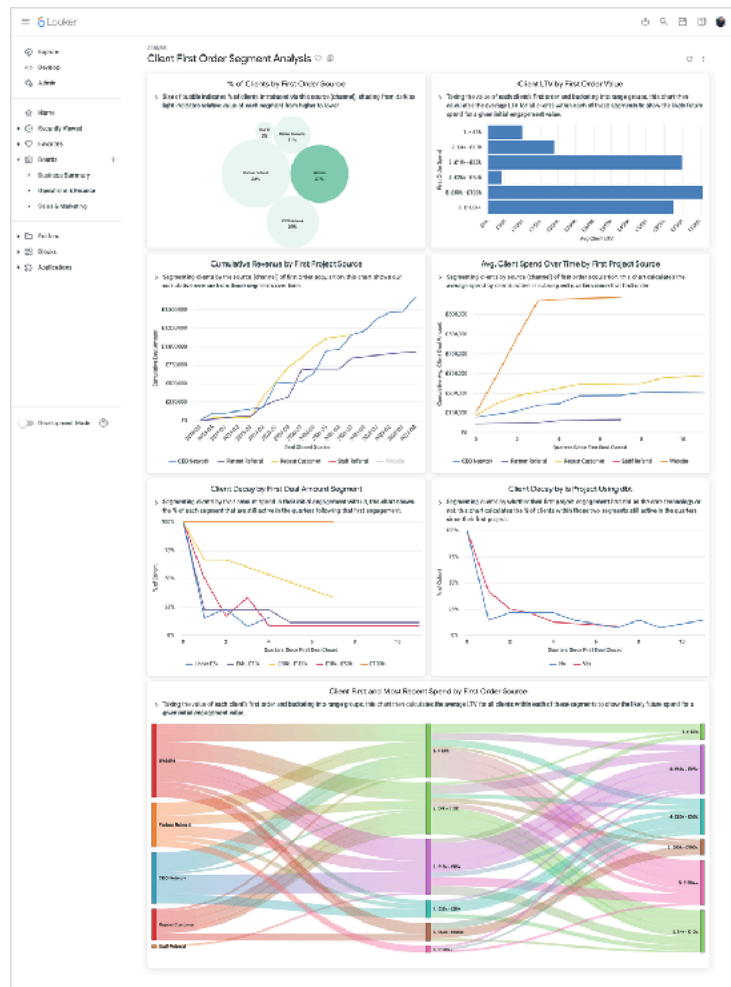
Along with **Recency, Frequency and Monetary Value (RFM)** segmentation, one of the most valuable and actionable ways that you can segment your customers is by the first, last and most frequent products and services they order.

The product a customer first orders from you is usually a good predictor of customer persona and, for example, their likelihood to go on and make more purchases.

The amount a customer spends on their first order is often a good predictor of the amount they'll spend with you over their lifetime.

Channels that bring in customers who over time become your most valuable customers are the channels you should focus your future marketing budget on.

We use first and last order customer segmentation to help us make decisions around which sales channels to invest marketing budget on, which products to base our technology recommendations on and in predicting the future lifetime value of new clients, as shown in this example Looker



dashboard.

Segmenting your customers by first order in a tool such as Looker is a relatively simple process if you're comfortable with SQL and Looker development, as it will typically involve adding two additional LookML views into your Looker project to calculate:

1. For each customer, the first and last order values for order elements that you're interested in – product category ordered, sales channel for that order, etc
2. For each transaction, the months, quarters and weeks that the transaction date was from that customer's first order

My usual method for creating these two LookML views is to first create a Looker query that returns all customer orders, one row per order, with all of the fields you want to segment by and also the customer and transaction keys, so that we can join the final LookML views back into our LookML model at the end.

Companies	Company Pk	Sales (Hubspot) Deal Closed Date	Sales (Hubspot) Deal source	Sales (Hubspot) Deal type	Sales (Hubspot) Partner referral	Sales (Hubspot) Pricing Model	Sales (Hubspot) Sprint Type	Sales (Hubspot) Number of Sprints	Sales (Hubspot) Deal Currency Code	Sales (Hubspot) License Internal (Yes / No)	Sales (Hubspot) Services (Yes / No)	Sales (Hubspot) Previews (Yes / No)	Sales (Hubspot) Cco (Yes / No)	Sales (Hubspot) Looker (Yes / No)
1	4c79191030c086002074614812444	2021-11-11		Existing Client		Fixed-Price Sprints Full	Data Engineering	0	GBP	No	No	No	No	No
2	1264e3ae70147125498015118a3032a	2021-11-11	Partner Referral	New Business	Partner Analytics	Fixed-Price Sprints Full	Data Analytics	0	GBP	No	Yes	Yes	No	No
3	174457922f911601e921620282af749	2021-10-31	Repeat Customer	Existing Client		Fixed-Price Sprints Full	Data Analytics	4	USD	No	Yes	Yes	No	Yes
4	80282968722109149106129743074	2021-10-31	Repeat Customer	Existing Client		Day Rate (Full)	Data Analytics	3	GBP	No	Yes	Yes	No	No
5	4933547ba7de6b0d6-7d5081451c6fd	2021-10-01		New Business		Fixed-Price Sprints Full	Data Engineering	1	USD	No	Yes	No	No	No
6	4933547ba7de6b0d6-7d5081451c6fd	2021-10-01	Website	Existing Client			Data Analytics	6	USD	No	Yes	No	No	No
7	14420531a813844034db88ef1d1df	2021-01-21	Repeat Customer	Existing Client		Fixed-Price Sprints Full	Data Analytics	6	USD	No	Yes	Yes	No	No
8	12496737c3d1986f1051401f3c89a58	2021-01-21	Repeat Customer	Existing Client	Looker	Fixed-Price Sprints Full	Data Analytics	1	GBP	No	Yes	No	No	No
9	14420531a813844034db88ef1d1df	2021-01-14	Website	New Business		Fixed-Price Sprints Full	Data Analytics	1	USD	No	Yes	Yes	No	No
10	1744f228f995e5c772a91ca2a5a2c2b	2021-01-10	Website	New Business		Fixed-Price Sprints Full	Data Analytics	1	USD	No	Yes	No	No	No
11	1898c0333a1c661a6507f6d5d236	2021-07-27	CEO Network	New Business		Fixed-Price Sprints Full	Data Analytics	6	GBP	No	Yes	Yes	No	No
12	12496737c3d1986f1051401f3c89a58	2021-07-05	Partner Referral	New Business	Looker	Fixed-Price Sprints Full	Data Analytics	1	GBP	Yes	Yes	Yes	No	No
13	0ba3141f036163138d1c1886d4e0e08	2021-01-04	Repeat Customer	Existing Client		Fixed-Price Sprints Full	Data Analytics	1	GBP	No	Yes	No	Yes	No
14	160c33b7b7a1841c83a95137561a110	2021-04-31	Repeat Customer	Existing Client		Fixed-Price Sprints Full	Data Analytics	3	GBP	No	Yes	Yes	No	Yes
15	408484050e8880526a481481344c	2021-04-31		Existing Client				0	GBP	No	No	No	No	No
16	1f624ae0f08144a74217a77f8efc	2021-04-01		Existing Client				0	EUR	No	No	No	No	No
17	80282968722109149106129743074	2021-04-01		Existing Client				0	GBP	No	No	No	No	No
18	4c79191030c086002074614812444	2021-04-01		Existing Client				0	GBP	No	No	No	No	No
19	1091486051d00501f878a439919a03	2021-01-01		Existing Client				0	USD	No	No	No	No	No
20	1f624ae0f08144a74217a77f8efc	2021-01-04	Repeat Customer	Existing Client		Day Rate (Full)		0	EUR	No	Yes	No	No	No

Then I'd click on the SQL button in the explore Data panel to show the SQL that Looker generated, which would typically look like the example below for a Google BigQuery data warehouse data source:

```

SELECT
  companies_dim.company_pk AS companies_dim_company_pk,
  (DATE(deals_fact.deal_closed_ts)) AS deals_fact_deal_closed_date,
  deals_fact.deal_source AS deals_fact_deal_source,
  deals_fact.deal_partner_referral AS deals_fact_partner_referral,
  deals_fact.deal_pricing_model AS deals_fact_pricing_model,
  ...

```

```

FROM `analytics.companies_dim` AS companies_dim
LEFT JOIN
  `analytics.deals_fact` AS deals_fact
ON
  companies_dim.company_pk = deals_fact.company_pk
WHERE
  (deals_fact.pipeline_stage_closed_won )
GROUP BY
  1,2,3,4,5...

```

To turn this into a query that calculates the first order value for those columns for each customer, I'd use the `FIRST_VALUE () OVER ()` analytic function to return those first order values and then wrap the query results in a `GROUP BY` aggregation to return just one row per customer, after first removing transaction date from the results, like this:

```

WITH deals AS
(
  SELECT
    companies_dim.company_pk AS companies_dim_company_pk,
    first_value(deals_fact.deal_source) over
(partition by companies_dim.company_pk order by
deals_fact.deal_closed_ts)
  AS is_first_deal_license_referral,
    first_value(deal_partner_referral) over
(partition by companies_dim.company_pk order by
deals_fact.deal_closed_ts)
  AS is_first_deal_services,
    first_value(deal_pricing_model) over
(partition by companies_dim.company_pk order by
deals_fact.deal_closed_ts)
  AS is_first_deal_dbt,
  ...
  FROM
    `analytics.deals_fact` AS deals_fact
  JOIN
    `analytics.companies_dim` AS companies_dim
  ON
    deals_fact.company_pk = companies_dim.company_pk
  WHERE
    deals_fact.pipeline_stage_closed_won
)
SELECT
  *

```

```

FROM
  deals
GROUP BY
  1,2,3,4...

```

Taking the same initial order-level Looker query and then reducing it to just the order key and transaction date, I can then use the FIRST\_VALUE analytic function along with the BigQuery DATE\_DIFF function to calculate months, quarters and years since each customers' first order date:

```

with transactions as (
  SELECT
    deals_fact.deal_pk,
    first_value(deal_closed_ts) over
(partition by company_pk order by deal_closed_ts) as
first_deal_closed_ts,date_diff(date(deal_closed_ts),
  first_value(date(deal_closed_ts)) over (partition by company_pk
order by deal_closed_ts),MONTH) as months_since_first_deal_closed,
  date_diff(date(deal_closed_ts),first_value(date(deal_closed_ts))
over (partition by company_pk order by deal_closed_ts),QUARTER) as
quarters_since_first_deal_closed,
  date_diff(date(deal_closed_ts),first_value(date(deal_closed_ts))
over (partition by company_pk order by deal_closed_ts),YEAR) as
years_since_first_deal_closed
  FROM
    `analytics.deals_fact` AS deals_fact
  WHERE
    deals_fact.pipeline_stage_closed_won
)
SELECT
*
FROM
transactions
;;

```

Note that for this second SQL query, we do not need to aggregate the rows it returns as we want to join it back into our Looker model at the individual transaction level.

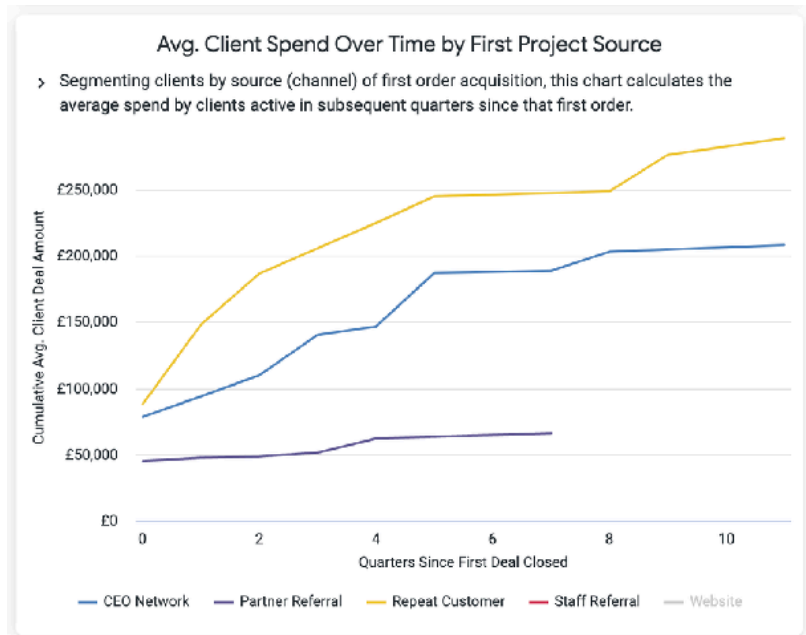
Finally I then create two new LookML views based on these queries as the source for their derived table SQL, and then join them back to the existing LookML

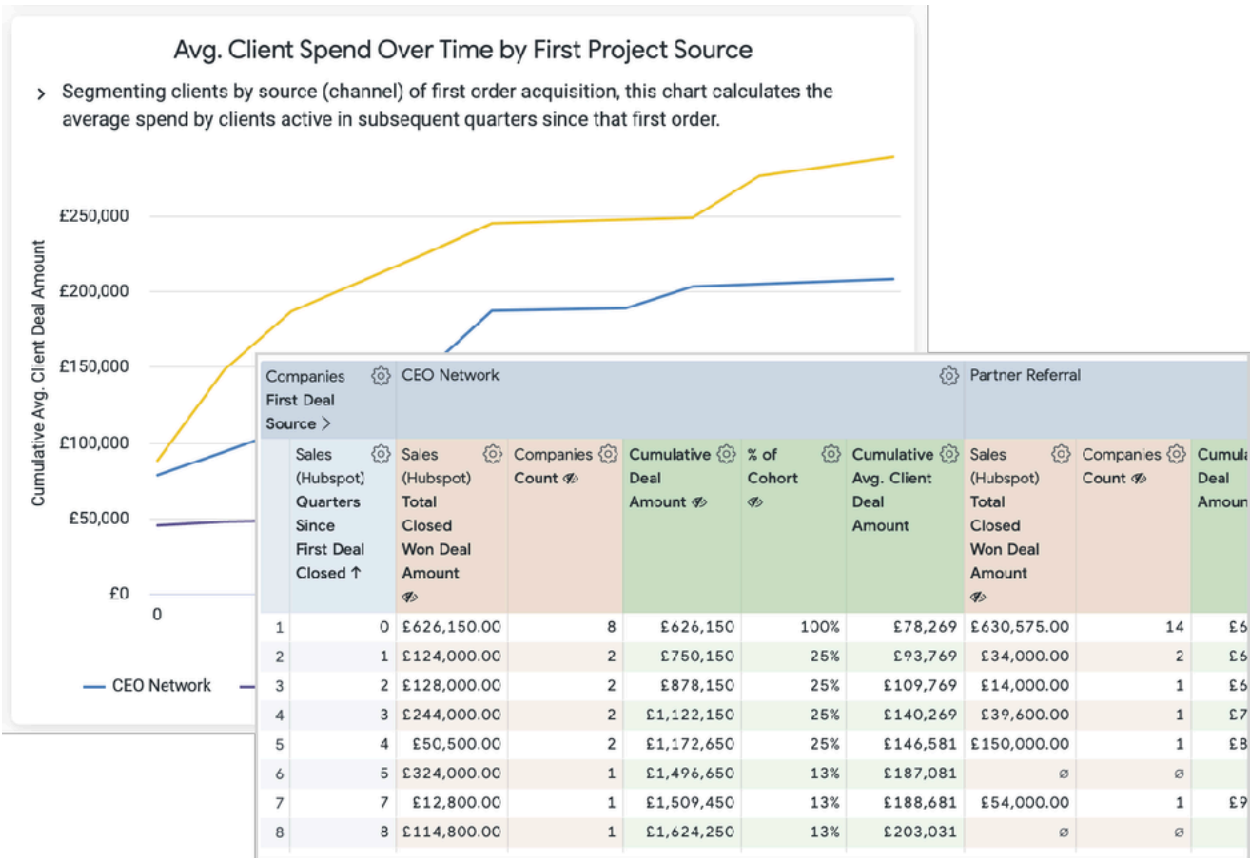
customer and transaction views within in my LookML model at the individual customer and transaction level, like this:

```
join: customer_first_deal_cohorts
{
  view_label: " Sales (Hubspot)"
  sql_on: ${deals_fact.deal_pk} =
${customer_first_deal_cohorts.deal_pk};;
  type: inner
  relationship: one_to_one
}
join: customer_first_order_segments
{
  view_label: " Companies"
  sql_on: ${companies_dim.company_pk} =
${customer_first_order_segments.companies_dim_company_pk} ;;
  type: left_outer
  relationship: one_to_one
}
```

.. and now I can start creating my Looker dashboard visualisations. Most of the charts are a variation on the one below which charts the average spend for each segment of customers indexed by quarters since their first order.

All of these visualizations use a set of common Looker table positional and running total calculations such as the ones in the example screenshot below, with each individual visualisation using particular selections of those calculations.





For example, another variation on this visualization are the Client decay visualizations in the main dashboard at the start of the article where the initial 100% population of each segment at the time of their first order is then recalculated each quarter after that first order to show the % still active (i.e. have ordered) after that time, using a table calculation such as:

```
(#{companies_dim.count} / index(#{companies_dim.count},1))
```

And finally, if you're wondering about the final visualisation showing customer spend on projects at the start and end of their engagement with us, it uses Sankey visualization type that you can install from the Looker Marketplace – useful also for customer journey-type data visualizations.

# Building up a Semantic Layer with dbt Metrics, Cube and Droughty

Olivier Dupuis, January 2023

I want to live in a world where data platform technologies can coexist together. And if they won't do it willingly, I'm happy to force their hand a little :) That's the situation I'm in when it comes to the semantic layer where I want dbt Labs and Cube to just be the BFF they're meant to be.

The analytics engineering world has been abuzz with the semantic layer in the past year. Beyond the hype, what is its purpose and how do you implement it? Let's go through a real-world example using the discursus project as a playground.

The image is a collage illustrating a semantic layer implementation. It features several key components:

- Data Model Diagram:** Shows a fact table `observations_fct` with columns `event_fk`, `observation_type`, `observation_page_description`, `observation_url`, `observation_page_title`, `observation_keywords`, and `published_date`. It also shows dimension tables `movements_dim` and `actions`.
- Dashboard:** Titled "North American Protests", it includes a line chart for "Events per Day" (with a peak of 262 events) and a pie chart for "Events per Protest Movement".
- Table View:** Displays columns such as `ACTION_GEO_COUNTRY`, `ACTION_GEO_FULL_NAME`, and `ACTION_GEO_LATITUDE`.
- Tooltip:** Explains that "Action geo latitude is a geographical coordinate system used for describing positions on the planet's surface. It is similar to standard latitude and longitude, except that it is measured in degrees, minutes, and seconds instead of decimal degrees. Action geo latitude helps determine a user's location, which can then be used to provide more accurate local information and services."



Gartner defines semantic data models as:

A method of organizing data that reflects the basic meaning of data items and the relationships among them.

(source: [Gartner glossary](#))

The components of a semantic layer are:

- Domain-specific entities, such as users and transactions
- Relationships between those entities
- Entity attributes, that can loosely be classified as either keys, dimensions, and measures/metrics
- Descriptions of the entities and their attributes

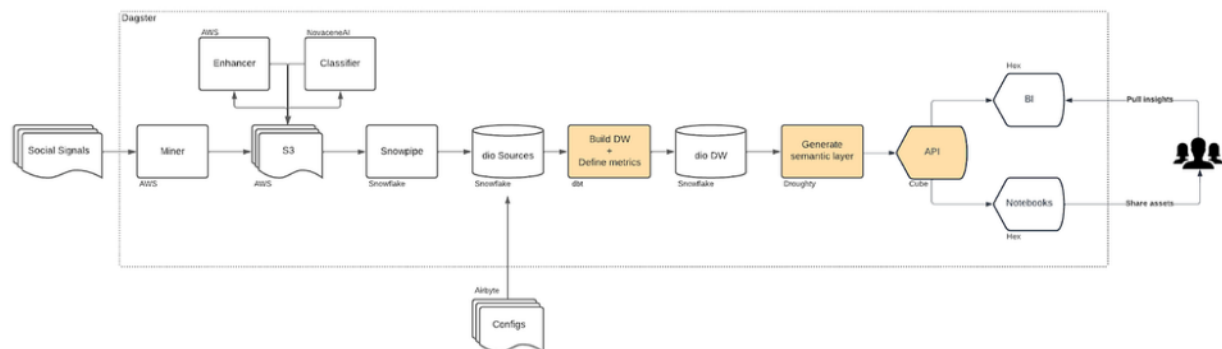
With those components in place, an end user is able to interact with entities and formulate domain-specific questions that return meaningful answers.

Let's work this all out through an example.

## Architecture

Building up a semantic layer requires that we start architecting those components into our data platforms and having a mechanism to surface them to end users.

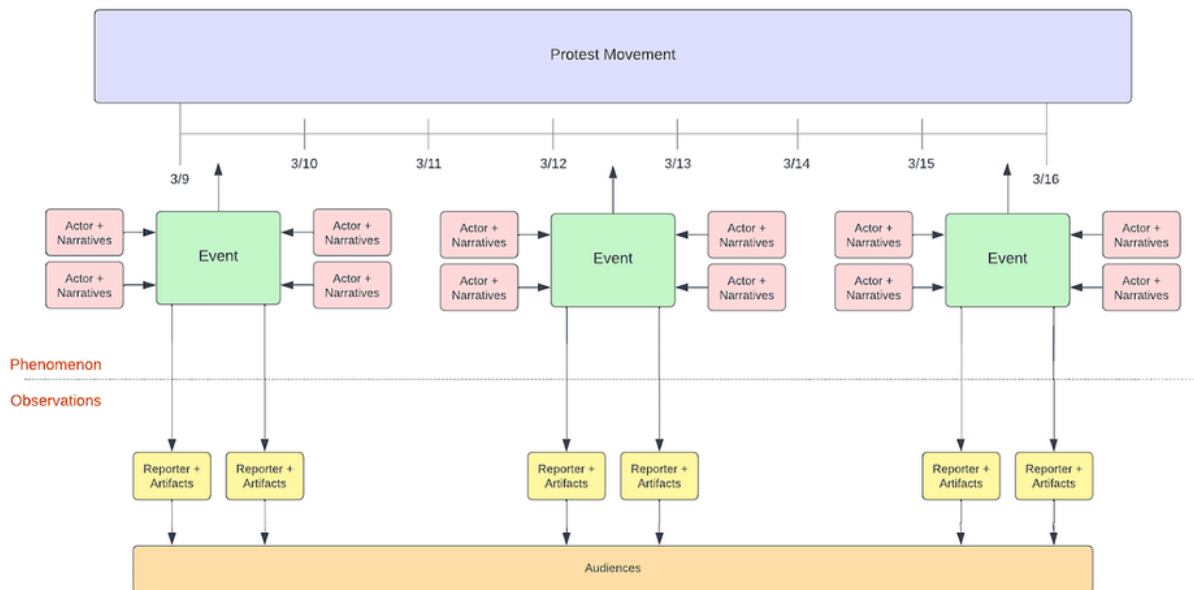
Going back to the discursus project, we have 3 key components in our architecture that play a role in serving the semantic layer:



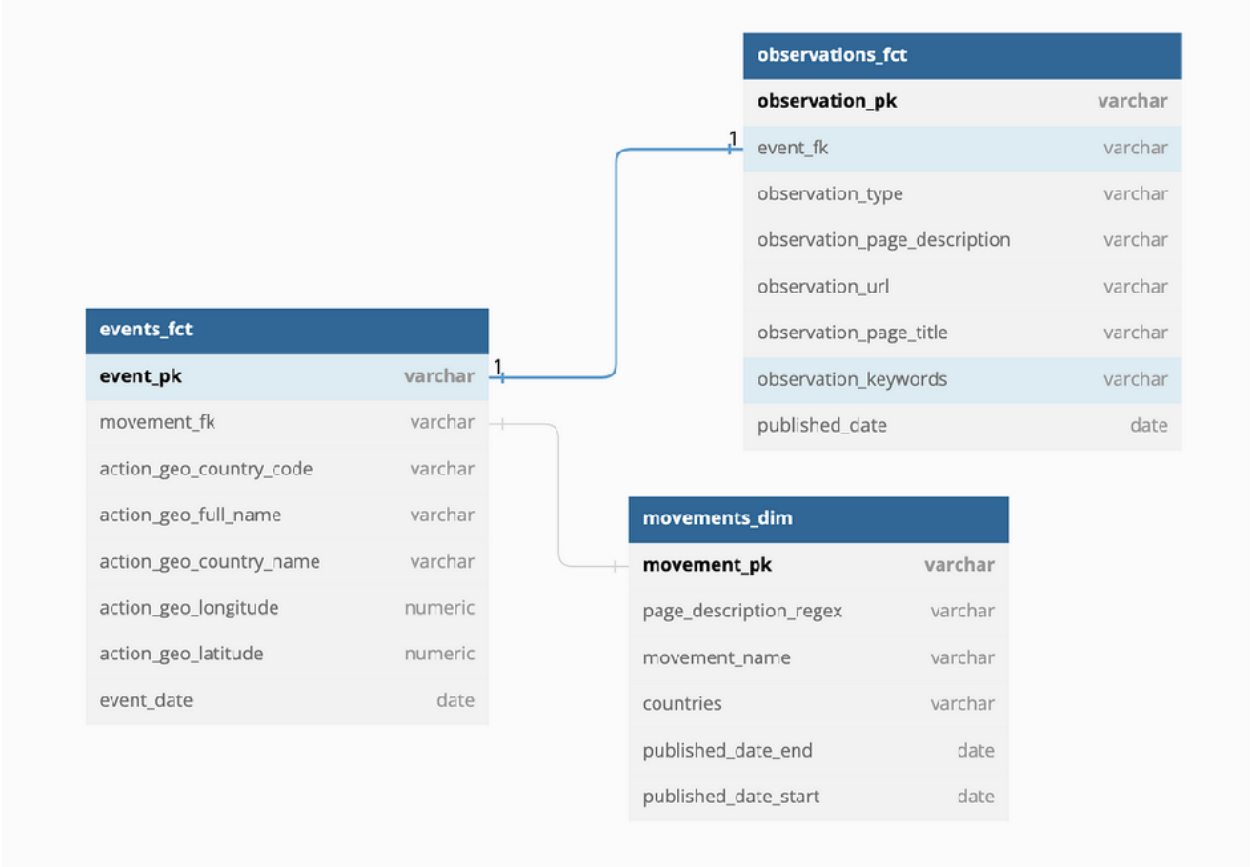
- **dbt**: to define entities, attributes, relationships and metrics
- **Droughty**: to generate semantic definitions
- **Cube**: to expose the semantic layer's components

## Defining your semantic layer

At the heart of our semantic layer is dbt. This is where we transform our raw data into a semantic structure that is a reflection of our domain, as shown in the diagram below:



and the discoursus entities, attributes and relationships, as shown in the entity relationship diagram below:



dbt is recognizing its role as the driving force in defining data’s semantic meaning. That meaning becomes in itself the API for other tools to interact with the data stored in a warehouse.

dbt started unveiling its semantic strategy with the [metric concept](#). As this is usually the main thing we interact with downstream (e.g. revenue, active users, inventory, etc.), it makes sense for dbt to provide a syntax to define those.

A very simple example using the discursus domain is the following:

```
version: 2

metrics:
  - name: event_count
    label: Events count
    model: ref('events_fct')
    description: "The count of events"

    calculation_method: count
    expression: event_pk

    timestamp: event_date
    time_grains: [day, week, month, quarter, year]

    dimensions:
      - movement_fk
```

This example simply tells downstream apps that our data domain includes a metric named `event_count` which is the count of events by daily, weekly, etc. time grains and which can be sliced by the `movement_fk` dimension. That definition sits on top of a model named `events_fct`. And so our DAG now includes a new metric node.



Notice that I said that the metric is defined on top of a model. But really, we should be saying that a metric is an attribute of an entity. In our example above, we would then be saying that `event_count` is the number of occurrences of the event entity.

And that is exactly what dbt Labs is working on as their next step of deploying the semantic layer. There is an open conversation in the [dbt-core git repository](#) about [introducing the entity concept](#) in their syntax. How that will manifest is beyond

the scope of this post, but it's a very interesting read for anyone curious about how that syntax is evolving.

## Serving your semantic layer

Although dbt is making interesting contributions to the semantic layer, and there is a [growing ecosystem of integrations](#), the semantic layer is fragmented. Different syntaxes and engines are available, but none has won the day yet.

It's not uncommon for a data platform to have a semantic layer with the following components:

- dbt: to define entities, attributes, relationships and field descriptions
- Looker (LookML): to provide self-serve BI capabilities for internal users
- Cube: to serve data to external-facing apps
- dbml: to document and share the latest state of a platform's data assets
- 

Each of those components has its own syntax. That is a lot of manual work to port your semantic definitions from one tool to another... and it's prone to errors.

That's where you need a tool like Droughty to build bridges between your semantic layer components.

Reading from the data warehouse's schemas, Droughty auto-generates semantic definition files.

For example, I can simply run this from the CLI:

```
droughty cube - profile-dir ./droughty_profiles.yml - project-dir ./droughty_projects.yml
```

And it will generate the following Cube definitions:

```
cube(`events_fct`, {
  sql: `select * from analytics.events_fct`,
  joins: {
    observations_fct: {
      relationship: `hasMany`,
      sql: `${CUBE.event_pk} = ${observations_fct.event_fk}`,
    },
  },
  dimensions: {
    action_geo_country_code: {
      sql: `action_geo_country_code`,
      type: `string`,
      description: `action geo country codes (agcc) are codes assigned to countries of the world by the action geo data company. the codes consist of five numbers and are used to classify countries by geographical region. action geo country codes are used in data analysis related to geospatial applications, including mapping, marketing, and population studies.`,
    },
    event_date: {
      sql: `event_date`,
      type: `time`,
      description: `event date is the day or days that a particular event is scheduled to take place.`,
    },
    event_pk: {
      primaryKey: true,
      type: `string`,
      sql: `event_pk`,
      description: `an event primary key is a unique identifier associated with an event, such as a conference, corporate gathering, or other special event. the primary key is typically assigned to each attendee and is used to help control event access and track attendance.`,
    },
  },
});
```

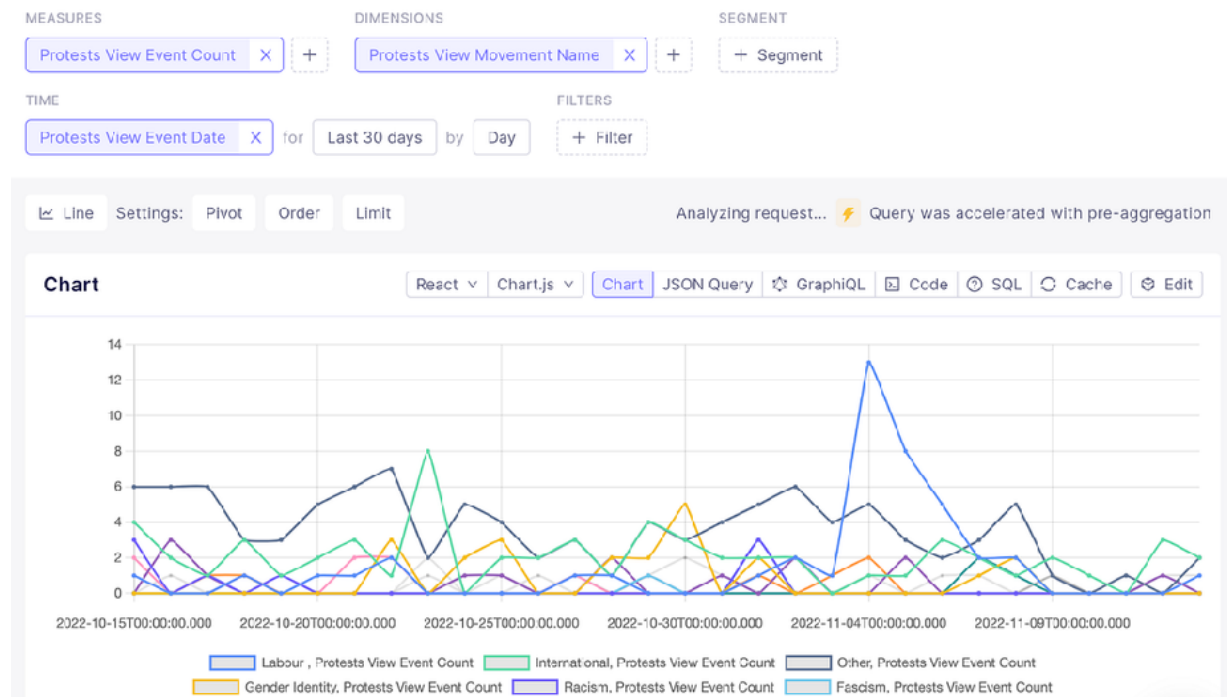
```

cube(`events_fct_extended`, {
  sql: `select * from ANALYTICS_QA.events_fct`,
  extends: events_fct,
  measures: {
    count_of_event_pk: {
      sql: `event_pk`,
      type: `count`,
    },
  },
})

```

Auto-generating semantic definition files accelerate the flow from modifying your semantic layer in dbt and exposing those changes to your semantic engines.

Below is an example of how I can now query my data warehouse with Cube using the semantic elements that were defined in dbt.



As [stated in a prior post](#), dbt is positioning itself to be the data platform standard. The dbt semantic syntax is promising and is worth integrating into data platforms. But that can't in itself drive an entire data platform's semantic layer.

We rely on other tools, which are well-established, mature and reliable. That includes Cube which is a great API solution for your data. It has its own syntax to define the semantic meaning of your data, but it shouldn't be a pain to derive its definitions from what has already been defined through dbt.

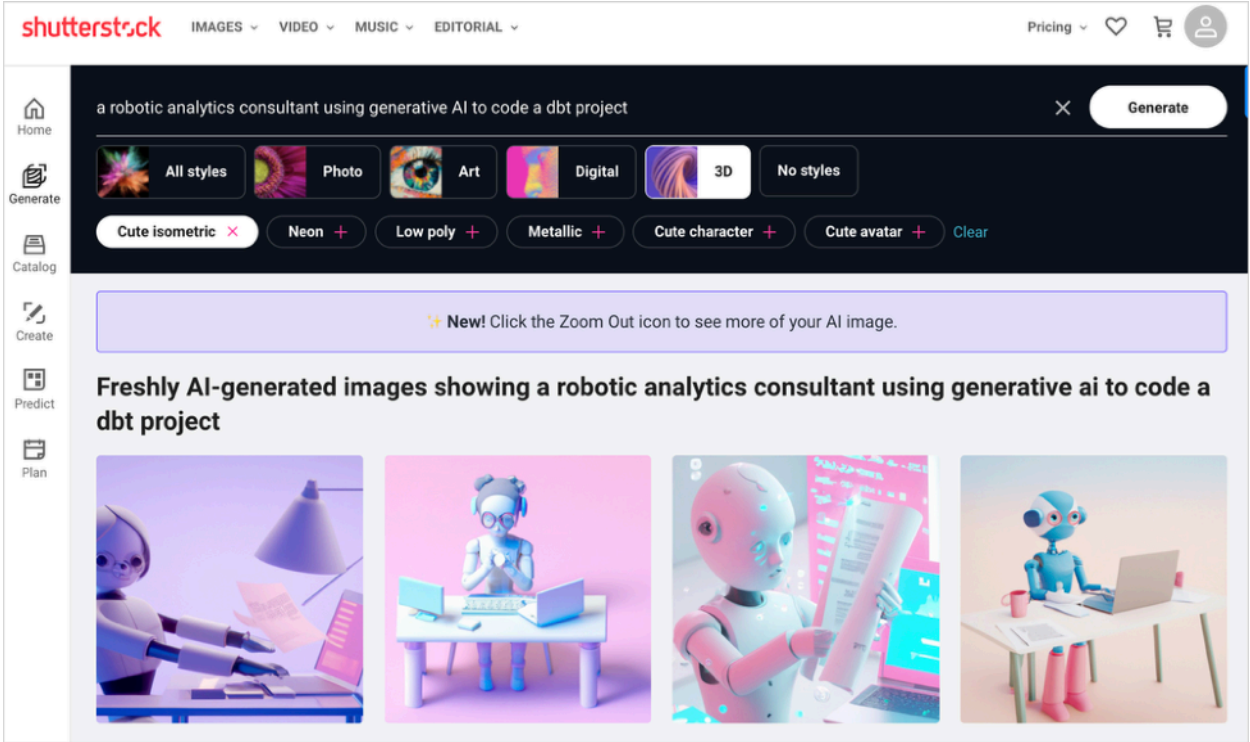
Tools like Droughty are necessary when you want to build a data platform that can use what's best from different ecosystems. Without formal integrations between certain tools, it's up to us data practitioners to force their hands a little.



# ChatGPT, Large Language Models and the Future of dbt and Analytics Consulting

Mark Rittman, March 2023

Everyone's talking about ChatGPT, generative AI models and the impact they are likely to have on [white collar jobs](#), software engineering (e.g. [Github Co-Pilot](#)) and more closer-to-home, [data roles within the modern data stack](#). We can even use generative AI to create an opening illustration for this blog, reassuring illustrators at least that their jobs are relatively safe for the time-being.



[Rittman Analytics](#) is a typical “boutique” modern data stack consultancy with 8 team members including myself, all of us hands-on practitioners working with data and clients every day and providing a range of services from strategy advice through to implementation and user enablement.

We typically have around half-a-dozen client projects running concurrently at any one time, spend quite a lot of time in face-to-face (usually virtual, these days) meetings with clients, project manage things on our side and take responsibility for making sure the overall project gets delivered as expected.

We're proud of our skills and testimonials from our clients ... but so were milkmen, bus conductors, high-street travel agents, [leech collectors](#), [toad doctors](#) and [gong farmers](#) just before their jobs became obsolete. So can ChatGPT-4, the latest version of OpenAI's Large Language Model interactive chat service, replace the role of an analytics engineer or even the need for an analytics consultancy?

I started a new ChatGPT session on the [OpenAI website](#) and give it a starting brief for an analytics project for a consulting business.



I have a google bigquery project and am working on a project for a consulting services business. I have used fivetran to replicate accounting data from xero into a dataset called "fivetran\_xero", CRM data from hubspot into a dataset called "fivetran\_hubspot" and project timesheet data from harvest into a dataset called "fivetran\_harvest". The common link between these datasets is the name of the client, which in Hubspot is a company name in Xero is a contact.

Create me a google bigquery schema to hold the combined set of data for these three sources, joining each on client name and with fact and dimension tables for the key business processes for a consulting business.

To create a schema for your Google BigQuery project that combines data from Xero, Hubspot, and Harvest, you can use the following design with fact and dimension tables tailored for a consulting services business.

First, let's create the dimension tables:

1. `dim_client`: This table will hold unique client information, with the client name being the common key across all three datasets.

```
sql Copy code
CREATE TABLE your_dataset.dim_client (
  client_id INT64 NOT NULL,
  client_name STRING NOT NULL,
  xero_contact_id STRING,
  hubspot_company_id STRING,
  PRIMARY KEY (client_id)
);
```

2. `dim_project`: This table will store project information, as projects are a key component of a consulting business.

```
sql Copy code
CREATE TABLE your_dataset.dim_project (
  project_id INT64 NOT NULL,
  project_name STRING NOT NULL,
```

ChatGPT goes straight into defining a reasonable starting schema, though it makes a basic SQL syntactical error right at the get-go. BigQuery does, in-fact, now support primary and foreign key constraints but doesn't enforce them, using them instead to provide additional schema metadata to BigQuery's query planner and the correct syntax for this new DDL feature is **PRIMARY KEY (column\_name) NOT ENFORCED**.

Another more subtle but, longer-term more significant, shortfall in ChatGPT's approach is that it only models the bare minimum table schema that would satisfy the literal request from the user, and not a more comprehensive schema that either experience, or further questioning, would lead you to create. But excepting the syntax error in each table's DDL, it does create a reasonable schema for the request it's been given.

I then ask ChatGPT to create a dbt package that takes the raw data from each of the Fivetran-sourced staging datasets, integrates and transforms it and then loads it into the fact and dimension tables it just defined a schema for.

Next I gave ChatGPT a more challenging task, to deduplicate client names not only when they are an exact match, but also when they just sound similar.

This often turns out to be a requirement when working with company names from a source such as Hubspot where reps input many variations on a single company's name when recording new sales deals, but you want those deals to resolve to a single company name when viewing that account's deal history.

now create a dbt package that will transform data in the harvest, xero and hubspot datasets, integrate it together and load it into the fact and dimension tables you just created

To create a dbt package for transforming and loading data from the Harvest, Xero, and Hubspot datasets into the fact and dimension tables, follow these steps:

1. Install dbt if you haven't already. You can do this by following the instructions in the dbt documentation: <https://docs.getdbt.com/dbt-cli/installation>
2. Initialize a new dbt project by running the following command:

```
cs:sharp Copy code  
dbt init your_project_name
```

3. Navigate to the newly created project directory:

```
...  
integrated_clients AS (  
  SELECT  
    row_number() over (ORDER BY c.client_name) AS client_id,  
    c.client_name,  
    h.hubspot_company_id,  
    x.xero_contact_id  
  FROM (  
    SELECT DISTINCT client_name  
    FROM hubspot_clients  
    UNION  
    SELECT DISTINCT client_name  
    FROM xero_clients  
  ) c  
  LEFT JOIN hubspot_clients h ON c.client_name = h.client_name  
  LEFT JOIN xero_clients x ON c.client_name = x.client_name  
)  
SELECT *  
FROM integrated_clients
```

Repeat this step to create transformation files for each table, such as 'dim\_project.sql', 'dim\_employee.sql', 'fact\_revenue.sql', 'fact\_expenses.sql', and 'fact\_timesheet.sql'. You can find examples of such transformations in the previous answer.

ChatGPT confidently offers to do this using BigQuery's Jaro-Winkler string similarity function, and rewrites the dbt code for me. Neat — I hadn't even heard of this function.

```
...
client_combinations AS (
  SELECT
    h.client_name AS hubspot_client_name,
    h.hubspot_company_id,
    x.client_name AS xero_client_name,
    x.xero_contact_id
  FROM hubspot_clients h
  CROSS JOIN xero_clients x
),

similar_clients AS (
  SELECT
    hubspot_client_name,
    hubspot_company_id,
    xero_client_name,
    xero_contact_id,
    JARO_WINKLER(hubspot_client_name, xero_client_name) AS similarity_score
  FROM client_combinations
),

best_match_clients AS (
  SELECT
    hubspot_client_name,
    hubspot_company_id,
    xero_client_name,
    xero_contact_id,
    RANK() OVER (PARTITION BY hubspot_client_name ORDER BY similarity_score)
  FROM similar_clients
  WHERE similarity_score > 0.85 -- You can adjust the threshold as needed
),

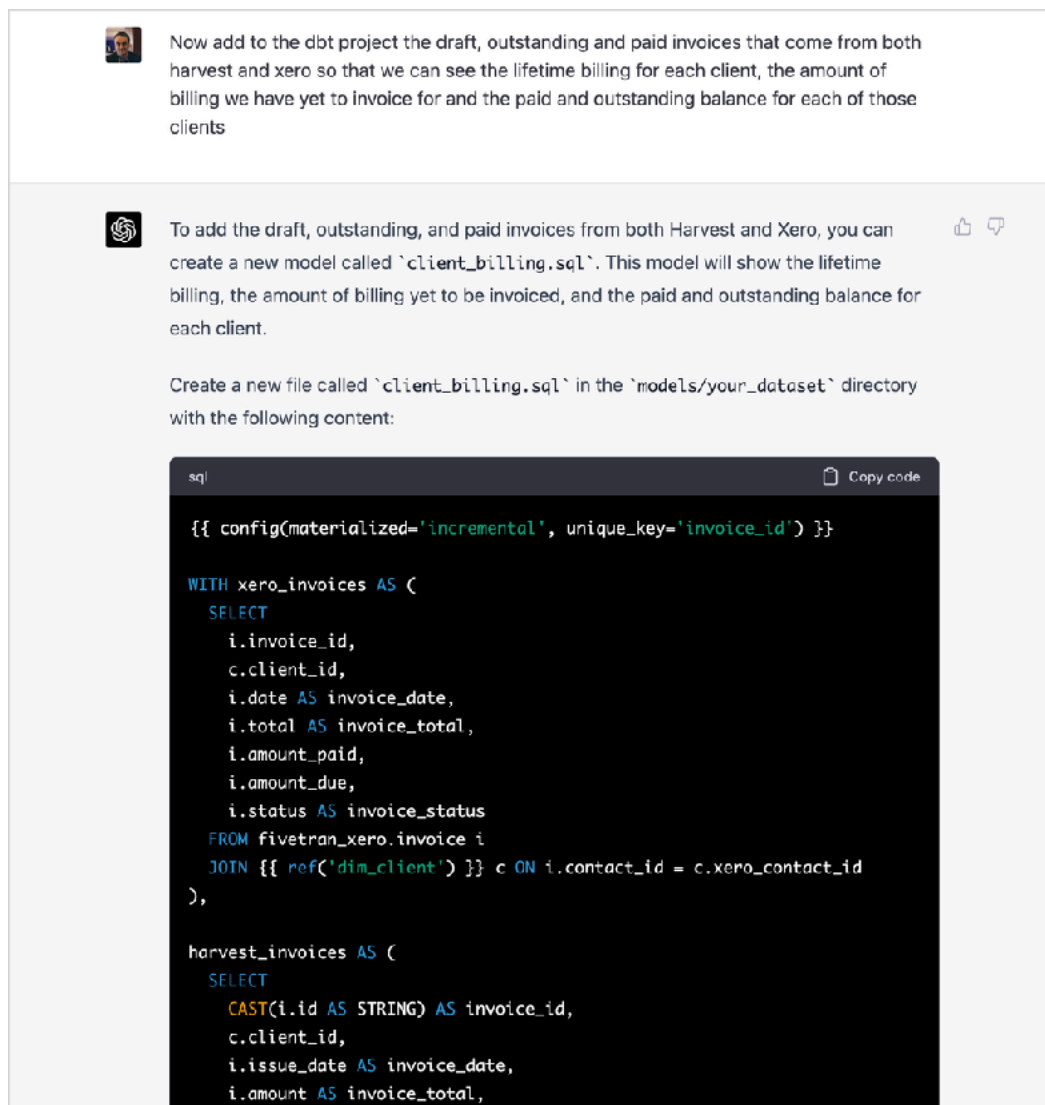
integrated_clients AS (
```

Except ... BigQuery doesn't have a JARO\_WINKLER string function. The [nearest you can get](#) at the time of writing this blog is to create a UDF (User-Defined Function) and the only database server that I'm aware of that has Jaro-Winkler functionality is Oracle Database and even then, it's a [parameter given to Oracle's UTL\\_MATCH function](#) rather than a SQL function in its own right.

ChatGPT in-fact seems to have invented its own BigQuery SQL function and then confidently given it to me as its solution, but this code just won't run and anyone handed it without BigQuery development experience just wouldn't have a clue what is wrong and what to do next.

I go on then to make further requests, and to be fair ChatGPT handles them well with a clear knowledge of what set of columns each source provides and how they map into common concepts such as invoices, invoice IDs and invoice dates as well as concepts such as paid, draft and outstanding invoices.

It also doesn't flinch when I ask it to add calculations for employee utilisation,



The screenshot shows a chat interface with two messages. The first message is a user request asking to add draft, outstanding, and paid invoices from both Harvest and Xero to a dbt project, with the goal of seeing lifetime billing, amount of billing yet to invoice, and paid/outstanding balances for each client. The second message is a response from ChatGPT, which provides instructions on how to create a new model named 'client\_billing.sql' and includes a SQL code block for the model's content. The code defines two CTEs: 'xero\_invoices' and 'harvest\_invoices', and uses them in a SELECT statement to retrieve invoice details from both sources, joined on client information.

Now add to the dbt project the draft, outstanding and paid invoices that come from both harvest and xero so that we can see the lifetime billing for each client, the amount of billing we have yet to invoice for and the paid and outstanding balance for each of those clients

To add the draft, outstanding, and paid invoices from both Harvest and Xero, you can create a new model called 'client\_billing.sql'. This model will show the lifetime billing, the amount of billing yet to be invoiced, and the paid and outstanding balance for each client.

Create a new file called 'client\_billing.sql' in the 'models/your\_dataset' directory with the following content:

```
sql Copy code

{{ config(materialized='incremental', unique_key='invoice_id') }}

WITH xero_invoices AS (
  SELECT
    i.invoice_id,
    c.client_id,
    i.date AS invoice_date,
    i.total AS invoice_total,
    i.amount_paid,
    i.amount_due,
    i.status AS invoice_status
  FROM fivetran_xero.invoice i
  JOIN {{ ref('dim_client') }} c ON i.contact_id = c.xero_contact_id
),

harvest_invoices AS (
  SELECT
    CAST(i.id AS STRING) AS invoice_id,
    c.client_id,
    i.issue_date AS invoice_date,
    i.amount AS invoice_total,
```

though in-practice we'd typically add a date spine table into the package to help calculate total capacity over all days, not just those on which an employee had actually worked.

I even ask ChatGPT to bring-in data from our HRMS system, [Humaans](#), suggesting that it uses that service's REST API to create a data extractor for data that it then



now update the dbt project to calculate utilisation at the monthly level for our project resources



To calculate the monthly utilization for project resources, you can create a new model called `monthly\_utilization.sql`. This model will calculate the total number of hours worked by each employee in a month and their billable hours, then compute the utilization percentage.



Create a new file called `monthly\_utilization.sql` in the `models/your\_dataset` directory with the following content:

```
sql Copy code
{{ config(materialized='incremental', unique_key='employee_id || month_start

WITH employee_monthly_hours AS (
  SELECT
    employee_id,
    DATE_TRUNC(DATE(entry_date), MONTH) AS month_start_date,
    SUM(hours) AS total_hours
  FROM {{ ref('fact_timesheet') }}
  GROUP BY employee_id, DATE_TRUNC(DATE(entry_date), MONTH)
),
employee_monthly_billable_hours AS (
  SELECT
    employee_id,
    DATE_TRUNC(DATE(entry_date), MONTH) AS month_start_date,
    SUM(hours) AS billable_hours
  FROM {{ ref('fact_timesheet') }}
  WHERE task_id IN (
    SELECT id
    FROM fivetran_harvest.task
    WHERE billable = TRUE
  )
  GROUP BY employee_id, DATE_TRUNC(DATE(entry_date), MONTH)
)
```

Regenerate response

loads into our BigQuery database. It then provides an overview of its solution and generates the Python code to connect to Humaan's REST API, download the relevant data and then load it into BigQuery using its Python client.



the client uses a hrms application called humaans to store the days off, sick days, date of recruitment and date of offboarding for each project resource. Humaans has a REST API but not a Fivetran packaged extractor. Tell me how we could use this data in Humaans to incorporate into the utilisation logic in your dbt package and make the utilisation calculations incorporate this additional data, update the dbt package to incorporate this new logic and write me an extractor to get the data from humaans, load it into the google bigquery project and transform that data as part of the dbt project you created for me.



To incorporate data from Humaans into the utilization logic, you'll need to extract the data using Humaans REST API and load it into Google BigQuery. You can then modify the existing `monthly\_utilization.sql` model to consider the additional data. Here's an outline of the process:

1. Write a Python script to extract data from Humaans API.
2. Load the extracted data into Google BigQuery.
3. Update the `monthly\_utilization.sql` model in the dbt project to incorporate the new data.

### 1. Extract data from Humaans API

First, you'll need to write a Python script to extract data from the Humaans API. You can use the `requests` library to make API calls and `pandas` library to process the data. Here's an example Python script:

```
python Copy code

import requests
import pandas as pd
import json

# Replace with your Humaans API key
API_KEY = 'your_humaans_api_key'

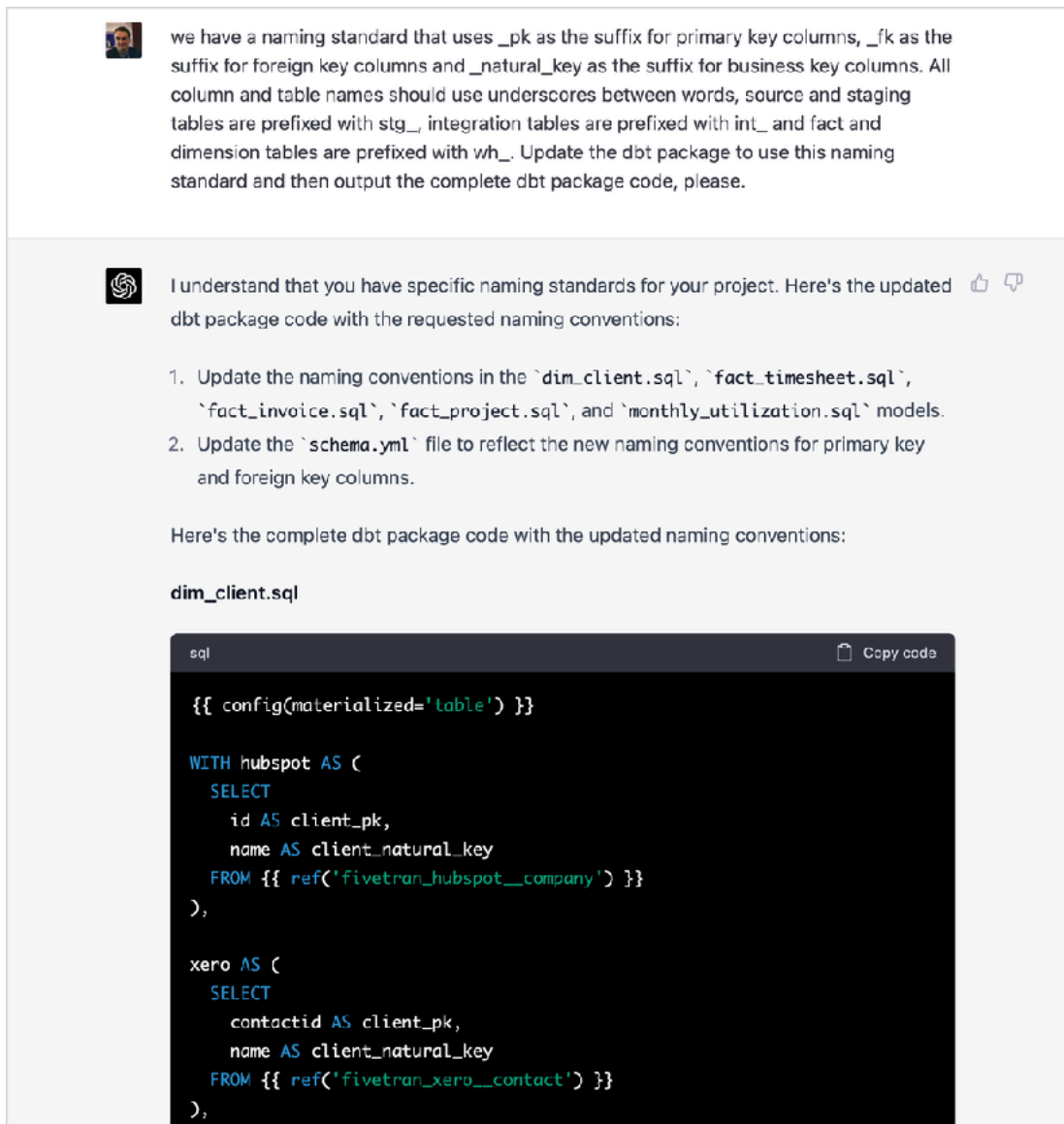
# API headers
headers = {
    'Authorization': f'Bearer {API_KEY}'
```

It's not the most robust solution and didn't consider how the Python scripts or indeed the dbt package would be scheduled and orchestrated, but it's a good initial prototype.

More significantly though if you're a freelancer taking on dbt work [on sites such as Fiverr](#) or a junior developer learning your trade by churning out dbt code tightly-specified by a solution architect, ChatGPT can most probably do that job better.





It can even incorporate our naming standards in the code it generates.



The screenshot shows a chat interface with a user and ChatGPT. The user asks for an update to a dbt package based on specific naming standards. ChatGPT responds with a list of instructions and a code snippet for a SQL model.


we have a naming standard that uses `_pk` as the suffix for primary key columns, `_fk` as the suffix for foreign key columns and `_natural_key` as the suffix for business key columns. All column and table names should use underscores between words, source and staging tables are prefixed with `stg_`, integration tables are prefixed with `int_` and fact and dimension tables are prefixed with `wh_`. Update the dbt package to use this naming standard and then output the complete dbt package code, please.

I understand that you have specific naming standards for your project. Here's the updated   dbt package code with the requested naming conventions:

1. Update the naming conventions in the `dim_client.sql`, `fact_timesheet.sql`, `fact_invoice.sql`, `fact_project.sql`, and `monthly_utilization.sql` models.
2. Update the `schema.yml` file to reflect the new naming conventions for primary key and foreign key columns.

Here's the complete dbt package code with the updated naming conventions:

`dim_client.sql`

```
sql  Copy code

{{ config(materialized='table') }}

WITH hubspot AS (
  SELECT
    id AS client_pk,
    name AS client_natural_key
  FROM {{ ref('fivetran_hubspot__company') }}
),

xero AS (
  SELECT
    contactid AS client_pk,
    name AS client_natural_key
  FROM {{ ref('fivetran_xero__contact') }}
),
```

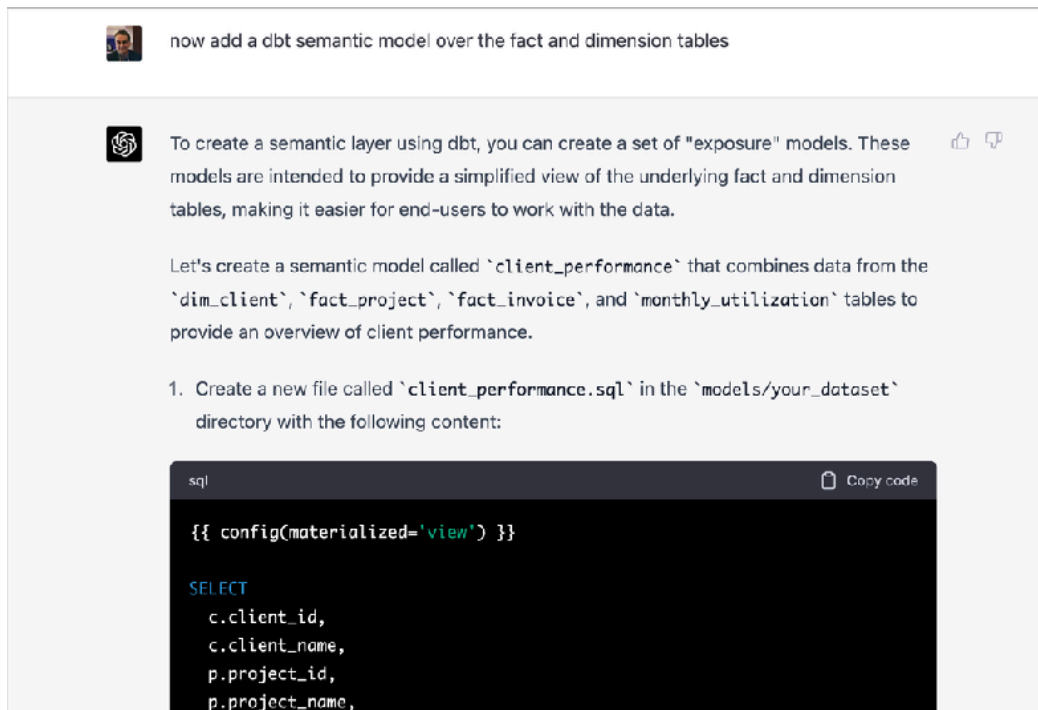
For now, although ChatGPT retains context from earlier instructions within the same overall chat session we'd need to provide details of these standards in any new chat session we initiated.

In the (near) future though it will be possible for consultancies and other organisations to create their own LLMs (Large Language Models) that [incorporate domain-specific training data](#) such as our corpus of dbt code sitting in private git repos along with our naming standards, standard operating procedures and



solution playbooks that together form the value and IP behind our service offerings.

Some things, such as more recent features added to dbt, will trip up a GPT model trained on data collected only up to a certain point in-time. When I ask it to add a [dbt Semantic Layer](#) over the models already created in the package it responds initially with an answer based on exposures, a pre-cursor to metrics and the semantic layer in dbt and then proposes a SQL view to provide a denormalized set of columns for that exposure.



now add a dbt semantic model over the fact and dimension tables


To create a semantic layer using dbt, you can create a set of "exposure" models. These models are intended to provide a simplified view of the underlying fact and dimension tables, making it easier for end-users to work with the data.




Let's create a semantic model called `client\_performance` that combines data from the `dim\_client`, `fact\_project`, `fact\_invoice`, and `monthly\_utilization` tables to provide an overview of client performance.

1. Create a new file called `client\_performance.sql` in the `models/your\_dataset` directory with the following content:

```
sql Copy code  
  
{{ config(materialized='view') }}  
  
SELECT  
  c.client_id,  
  c.client_name,  
  p.project_id,  
  p.project_name,
```

I then correct it and suggest a solution that uses the metrics feature introduced with more recent versions of dbt, to which ChatGPT responds first with an apology and then a revised solution that's not a bad start to incorporating metrics and a semantic layer into the package.

 no i meant a dbt metrics layer as you now get from dbt version 1.0.0 onwards. Please correct your response to the last question by using the new "metric" definition within dbt core

 I apologize for the confusion. In dbt 1.0.0 and onwards, you can define metrics to create a semantic layer for your data.  

1. Create a new directory called `metrics` within the `models` directory.
2. Create a new file called `client_metrics.yml` within the `models/metrics` directory.
3. Define the metrics in the `client_metrics.yml` file:


```
yml Copy code
version: 2

metrics:
  - name: total_revenue
    model: ref('fact_project')
    description: Total revenue for each project
    aggregation: sum
    column: total_revenue
```


Where this does fall short in-terms of a response you'd expect from a [dbt Preferred Partner such as ourselves](#), though, is that there's much more to consider here than just the syntax of the commands to implement this particular feature.

By using dbt's Semantic Layer you're implicitly making the choice to limit yourself to those BI and other tools within the modern data stack ecosystem that are compatible with dbt's Semantic Layer, Proxy Server and other enabling technologies; similarly, if you instead choose alternatives such as Looker's Universal Semantic Model or Cube's currently more feature-rich, but also more niche, semantic model then you're implicitly making choices about the wider solution that won't be apparent to a client who knows little about this area of technology.

And it certainly handles the relatively simple, or at least relatively easy to define, tasks that are put to it, and doesn't even try to conveniently forget to do the housekeeping tasks such as writing package documentation that takes time away from coding and other more interesting work.



For the Bigquery-compatible version of this dbt project, write a summary document in the form of a `readme.md` file for the client that provides an overview of the package, the data model for the final set of source, integration and warehouse tables and instructions on how to use the package.



Create a `README.md` file in the root directory of your dbt project and add the following content:

---

## Consulting Services Analytics dbt Project

This dbt project provides an integrated view of your consulting services business data, combining information from Xero, Hubspot, Harvest, and Humaans. The project contains a set of transformations to process and integrate data, as well as a set of metrics to provide a semantic layer for analysis and reporting.

### Data Model Overview

The data model consists of three layers:

1. **Source Tables:** Raw data replicated from Xero, Hubspot, and Harvest using Fivetran, and Humaans data fetched via a custom extractor.
2. **Integration Tables:** Intermediate tables that perform data transformations, clean up, and integrate data from multiple sources.
3. **Warehouse Tables:** Fact and dimension tables that contain the final, integrated data for analysis and reporting.

### Source Tables

- `fivetran_xero.contact``: Xero contact data (client information)
- `fivetran_hubspot.company``: Hubspot company data (client information)
- `fivetran_harvest.timesheet``: Harvest timesheet data (project timesheets)
- `humaans_data``: Humaans data (employee days off, sick days, date of recruitment, and offboarding)

But it makes mistakes, and howlers such as the JARO\_WINKLER one in this project, and only does what you ask it to do and no more. As such it's essentially another junior developer, full of enthusiasm, opinions and [plausible bullshit](#) that if supervised and instructed properly can multiply the productivity of a senior consultant but you wouldn't dare leave alone with a client to advise on their data analytics strategy.

It's even more of a dangerous prospect in the hands of an otherwise prospective client who knows what they want, thinks they know how to do it and just needs to find someone who can just do what he or she asks. These are often the most dangerous (but also entertaining and open-minded) clients who more than anything need saving from themselves and steering into the solution that actually meets their needs rather than delivering exactly what they asked for and it blowing-up shortly thereafter.

Make no mistake – future iterations of generative AI, public services such as ChatGPT and domain-specific versions on these underlying models will make IT and computing to-date look like the spinning jennys that were the start of the industrial revolution.

But there's more to delivering a successful, valuable and scalable modern data stack implementation than just writing dbt, Looker and Python code, starting with defining the actual problem to be solved and ensuring that what gets delivered makes sense, solves the client's problem and enables the types of business transformation and innovative services described by businesses such as [Breakthrough](#), [Conversion Interactive Agency](#) and [Torticity](#) in new client testimonials just added to our website.

# Building a Mobile-Friendly KPI Dashboard using Looker Studio Integration with Looker Universal Semantic Model

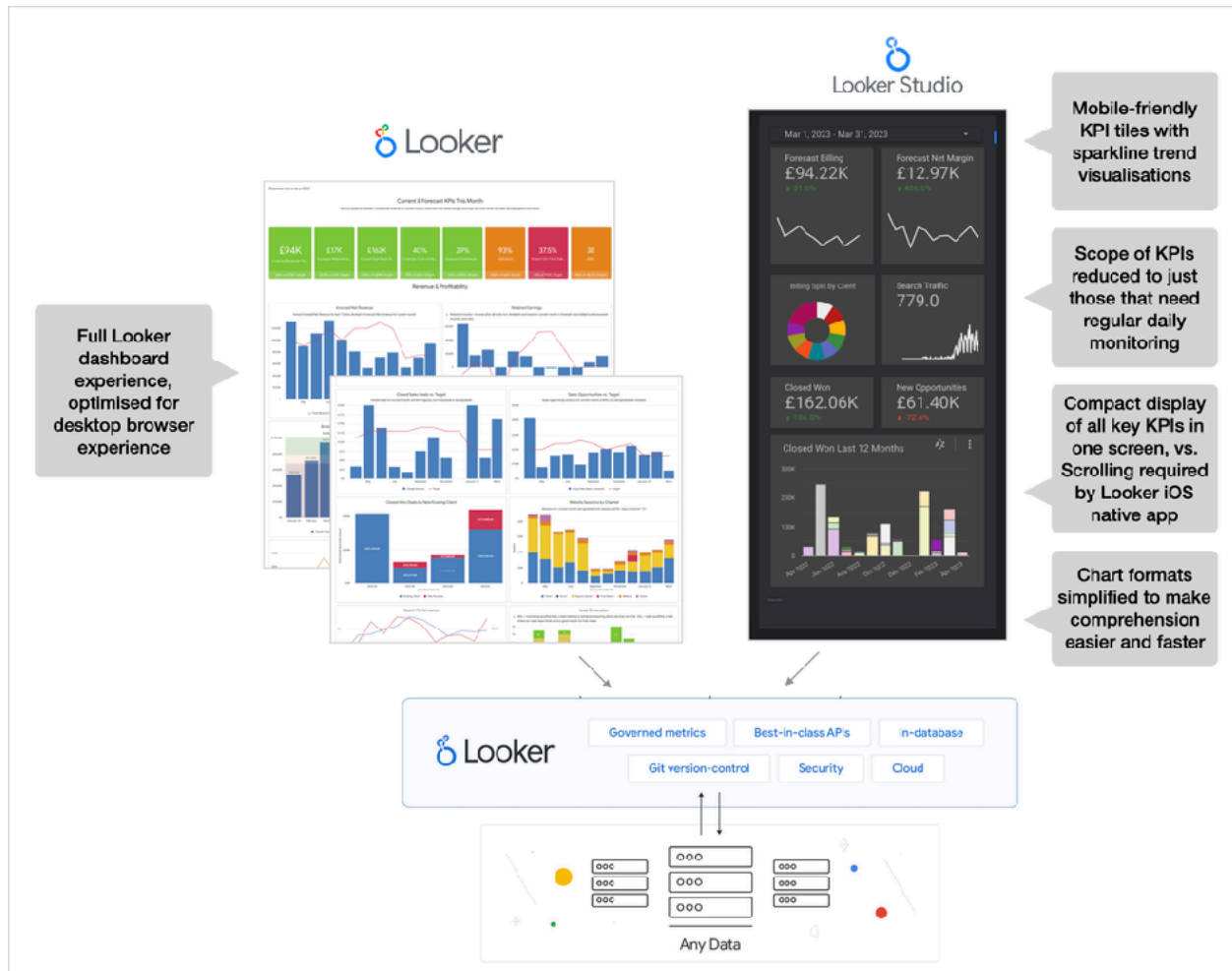
Mark Rittman, April 2023

One of the new features added into Looker in the past year is [integration with Looker Studio](#), the [new name](#) for Google Data Studio, along with similar integration with Google Sheets.

Users of Looker Studio can now build reports using governed metrics and dimensions from their [Looker semantic model](#) while Looker developers now have access to a much richer and flexible reporting tool should they want to go beyond standard Looker dashboards.

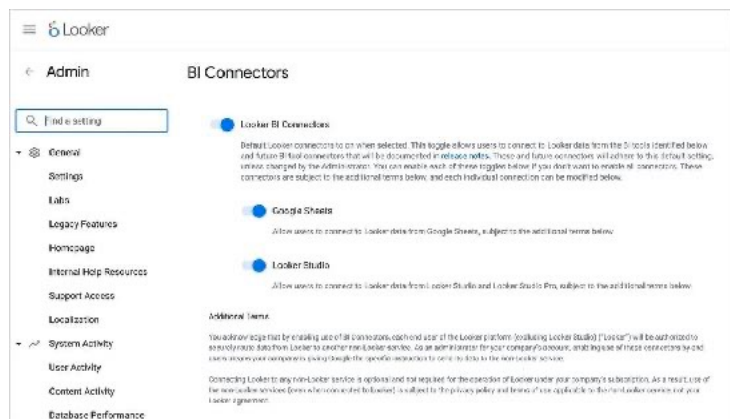
Google Cloud Platform recently announced BI Connector integration for [ThoughtSpot](#) and [Sisu](#) to go with the previously-announced [Tableau](#), together with a new forthcoming tool called Looker Modeler that enables LookML development of what's now termed the Looker Universal Semantic Model independent of the Looker web IDE.

For now though, one way that we've used this new integration at Rittman Analytics is to create a mobile-friendly web app KPI dashboard to accompany our [main Looker KPI dashboard](#), showing in a compact and mobile-friendly format the most important business metrics for our management team and based-on the same "single version of the truth" as the main dashboard.



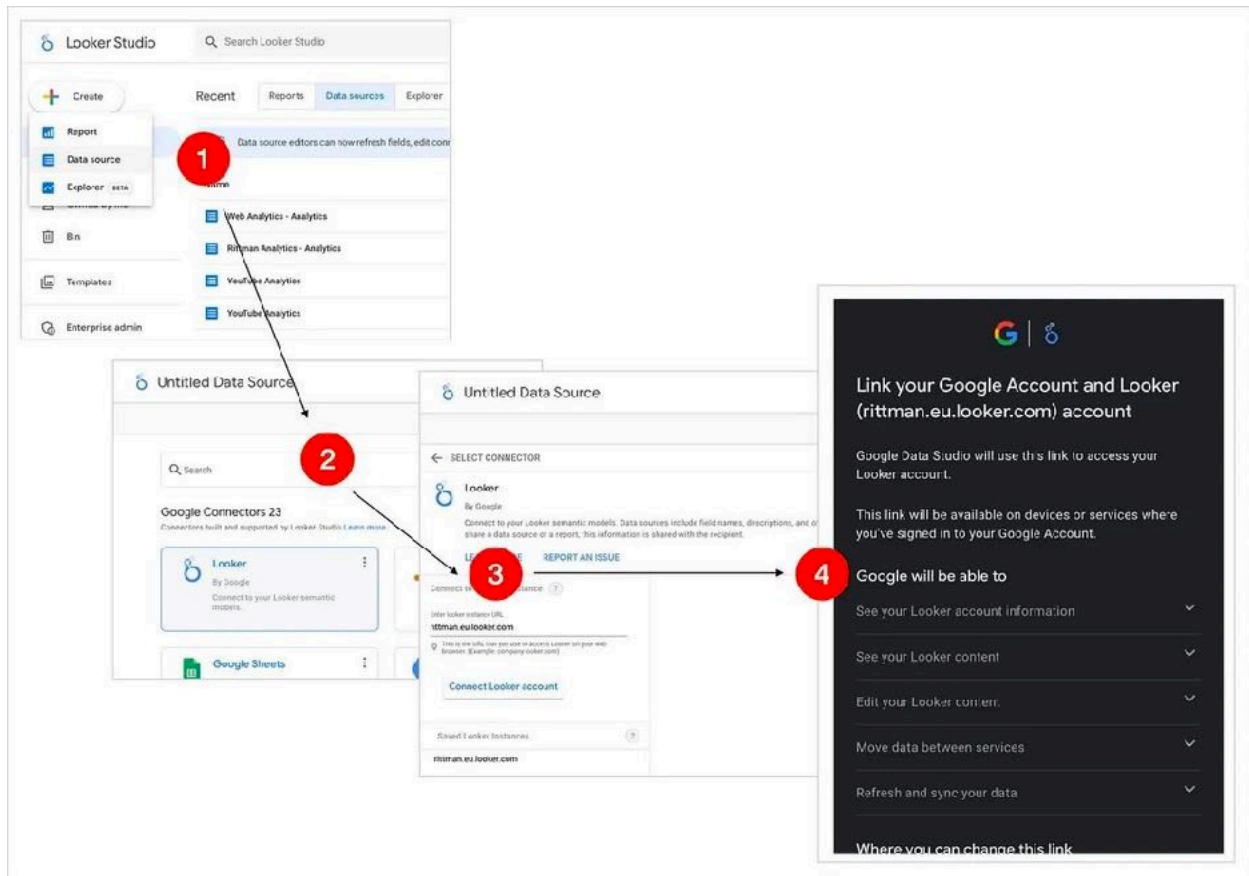
A pre-requisite for connecting Looker Studio to your Looker account is to first enable the BI Connectors feature within Looker's Admin menu.

Now, assuming that you've got an explore set-up within Looker that you want to use as



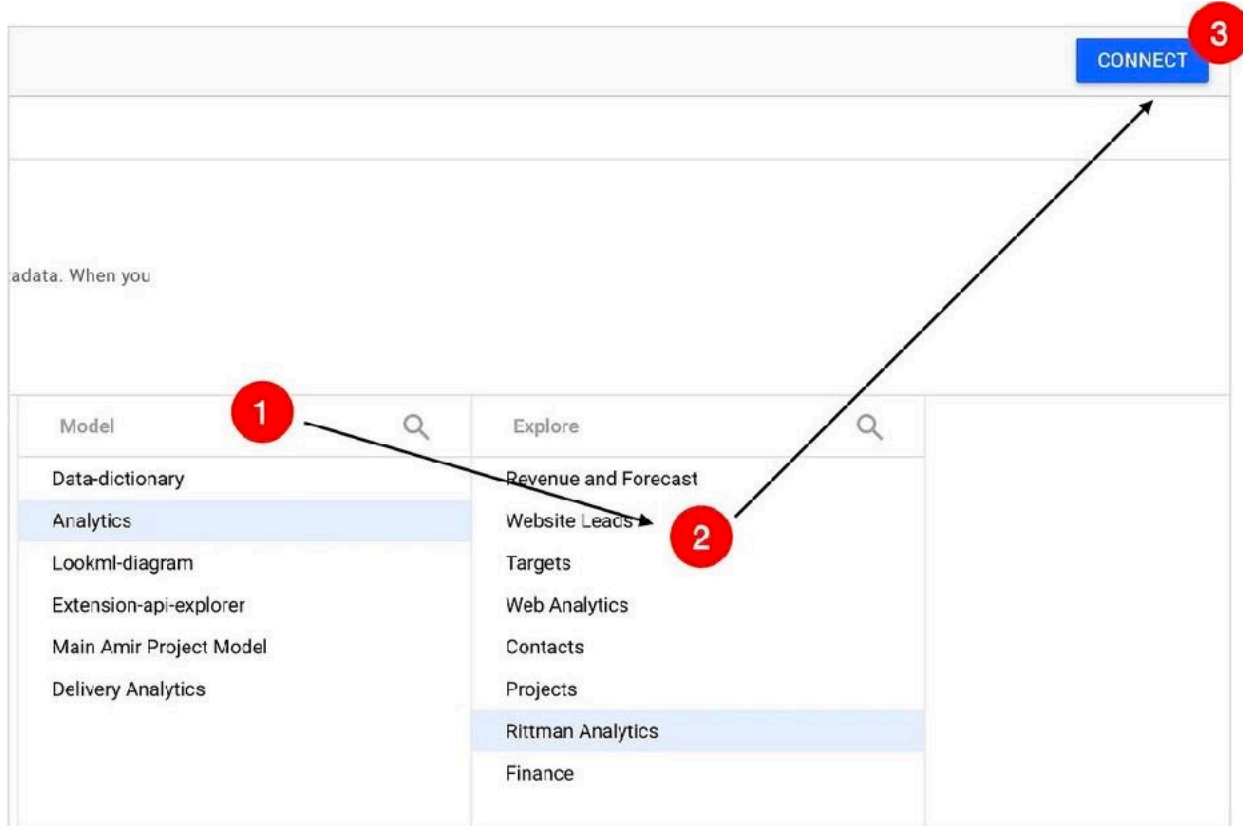
your Looker Studio data source, use your web browser to navigate to the Looker Studio [Data Sources](#) page.

From there, click the Create button and select Looker as the data source type, then enter the domain name for your Looker account and finally the Connect Looker Account button to authorise and create your connection.

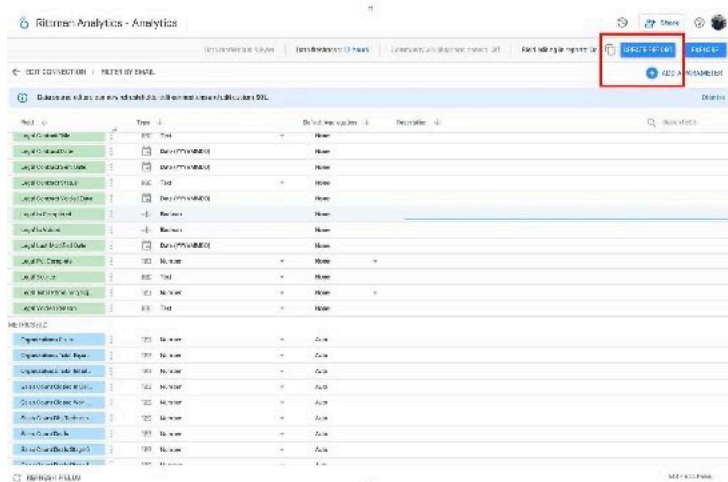


You should now see a list of models that your Looker account makes available. Select the model, and then the explore within that model that contains the data you are looking for, and then press the Connect button.

Note that, similar to the [Merge Query](#) feature within Looker's explore user interface, later on we can merge in additional Looker data sources into our report as well as data sourced from any of the other [official](#) and [community supported](#) data sources available for Looker Studio.



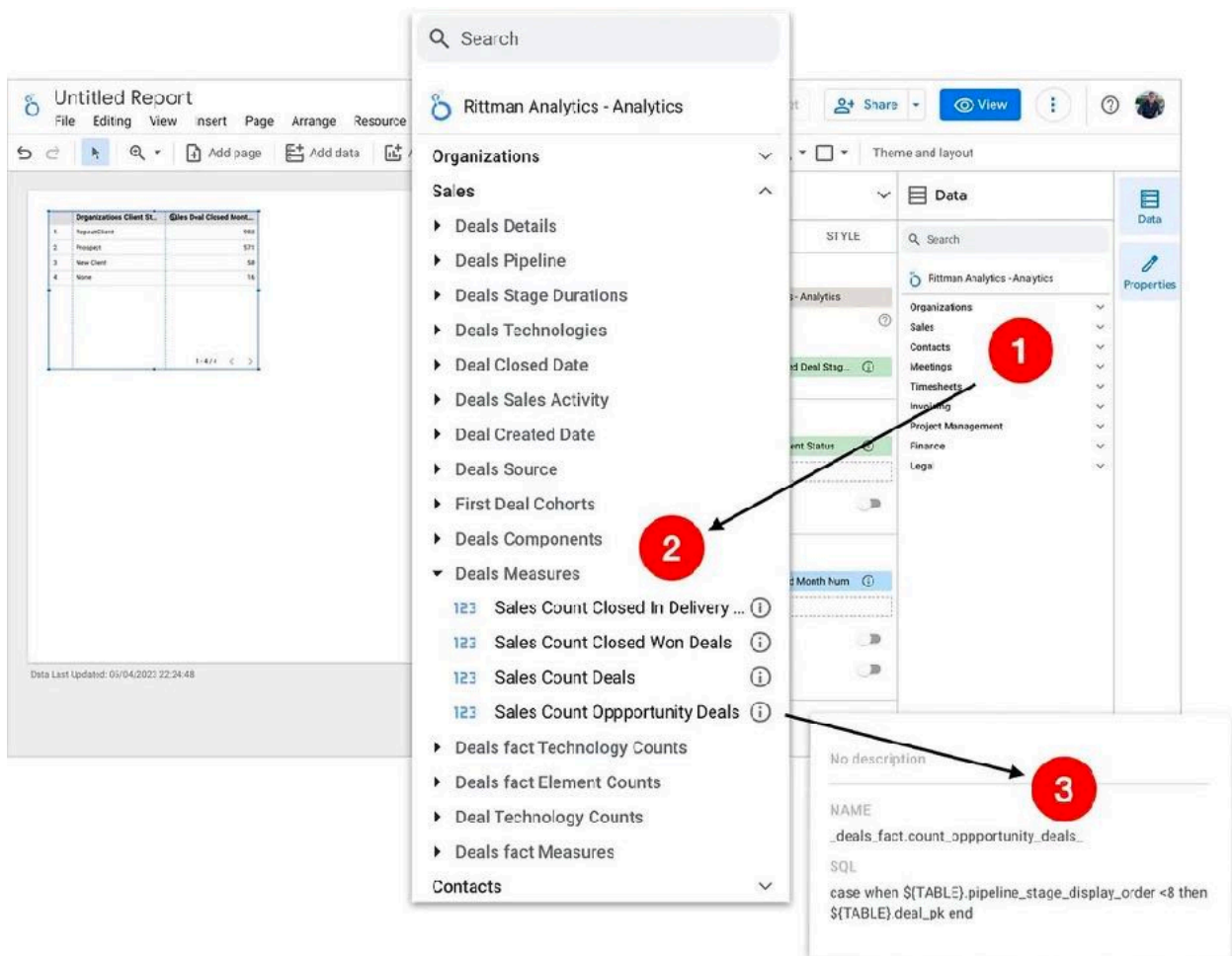
If you're familiar with the Data Studio report creation process you'll recognise the page where all of the dimensions and measure columns from your data source are presented and available to hide, format or add more derived columns for.





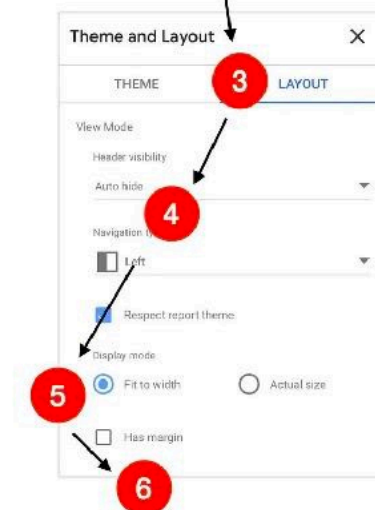
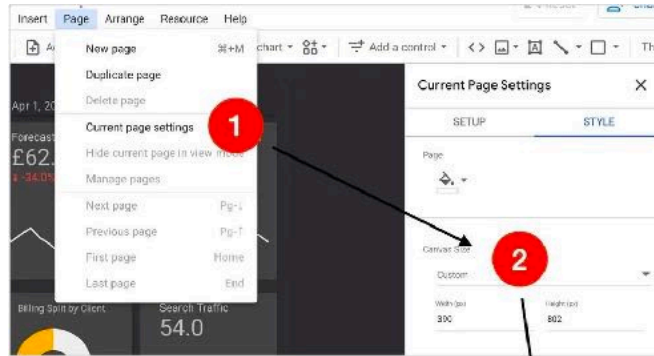
In the case of Looker-sourced measures you don't need to set their aggregation method, just leave them as Number as the LookML definition of those measures' aggregation type will automatically be used when displayed in a report; just press Create Report to start creating your report layout.

Creating a Looker Studio report with Looker as the data source adds a folder-style grouping of the views, dimensions and measures in your chosen explore, and you can even click on the information icon to see the LookML definition for each of the explore items.



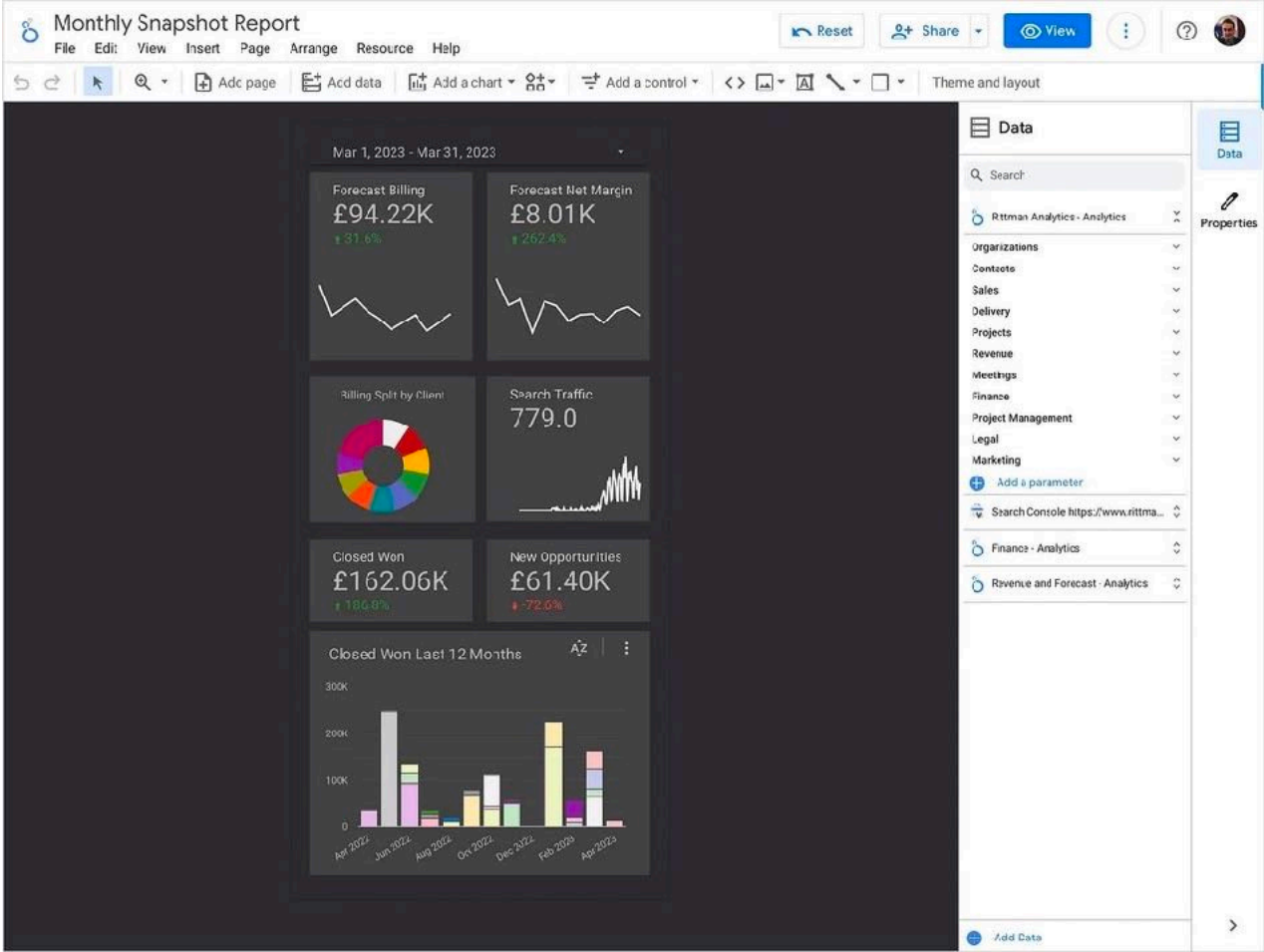
To make my Looker Studio report work as a web app-style mobile dashboard I then make the following configuration changes to a standard report template:

1. Select Page > Current page settings
2. Set the Canvas Size to 390 (px) Width and 802 (px) Height, the typical viewport screen area for a modern iPhone.
3. Now press the Theme and Layout button in the report header menu and select Auto hide as the Header visibility setting.
4. Set the Display Mode to Fit to width
5. Finally, uncheck the Has margin checkbox



Now the report template is all set to be mobile-friendly and formatted correctly, I then create my mobile KPI dashboard using dark mode colours, sparkline charts and stripped-back bar charts and KPI tiles that call-out our most important performance metrics.

Notice the three additional data sources that were added to the final report to create the final mobile KPI dashboard shown in the screenshot above; two more Looker explores that contained the Net Margin and Forecast Billing KPIs, and the [Google Search Console](#) data source for the Search Traffic KPI.



Using Looker Studio data source connectors to blend-in data from marketing, advertising and SaaS data sources is a useful lightweight alternative to using tools such as Fivetran and dbt to extract and then integrate those data sources into your warehouse.

The downside is that any integration of those data sources is specific to just that one report, but if all you want to do is display your ad spend data alongside other marketing metrics then this approach gets the job done without requiring IT support and the more complete integration I [blogged-about a few posts ago](#).

# Optimizing Data Materialization Using Dagster's Policies

Olivier Dupuis, April 2023

**M**anaging an effective asset materialization strategy for the discursus project, a data product on protest movements in North America, has proven challenging. As with most projects, some assets need to be materialized at different intervals or based on different criteria. For example:

- Some assets should run every 15 minutes on the mark.
- Others need to run immediately after an upstream asset materializes.
- While others should run only once their data becomes stale.

Fortunately, Dagster has launched a suite of APIs to declaratively define when an asset should be materialized.

This article will discuss my own experimentation with asset materialization policies based on my project's needs. Please refer to the official documentation and community channels to implement the right strategy for your own assets.

## Controlling Materialization Through Policies

Dagster allows you to control different asset materialization requirements using two types of policies:

1. Freshness policies — “specifies how up-to-date you want a given asset to be”. For example, I might want an asset to have data as fresh as its upstream dependencies within 15 minutes.
2. Auto-materialization policies — “automatically materialize assets when criteria are met”. For example, I might want to immediately materialize an asset once upstream assets have been materialized (eager).



In a pre-policy setup, I would have had schedules to materialize those assets. I would first define jobs:

```
# Packaging assets into a job
source_and_classify_relevancy_of_gdelt_assets_job = define_asset_job(
    name="source_and_classify_relevancy_of_gdelt_assets_job",
    selection=[
        "gdelt_events",
        "gdelt_mentions",
        "gdelt_mentions_enhanced"
    ]
)
```

And then schedule those jobs:

```
# Scheduling a job run
ScheduleDefinition(
    job=source_and_classify_relevancy_of_gdelt_assets_job,
    cron_schedule="47 * * * *"
)
```

This approach is not much different from other schedule-based orchestrators. You would need to think in terms of pipelines and the sequence of assets that require materialization, as well as how to optimize those jobs to minimize resource drain.

## Software-Defined Assets and Freshness Policies

The release of Dagster version 1.1 significantly changed our perspective on data platforms as pipelines, as it introduced the concept of software-defined assets. Assets became the first-class citizens of data platforms, rather than pipelines.

This shift meant that our responsibility was now to define each asset's freshness policy, and Dagster would take care of running jobs that would enforce those policies.

For example, if I have an asset with the following configurations:


```
@asset(
    non_argument_deps = {"gdelt_mentions_enhanced"},
    description = "Entity extraction of GDELT mentions",
    key_prefix = ["gdelt"],
    group_name = "prepared_sources",
    resource_defs = {
        'novacene_resource': my_resources.my_novacene_resource,
        'snowflake_resource': my_resources.my_snowflake_resource
    },
    freshness_policy = FreshnessPolicy(maximum_lag_minutes=60)
)
def gdelt_mentions_entity_extraction(context):
    # Bla
```

Dagster will interpret this policy as follows:

---

**Freshness policy** ▼

---

 At any point in time, this asset should incorporate all data up to 1 hour before that time.

---

By defining and enabling an asset sensor, Dagster would periodically assess an asset's freshness:

- Fresh: The asset has the latest data
- Stale: The asset still conforms to the freshness policy, but there is more up-to-date data in upstream assets
- Overdue: The asset's data no longer conforms to the freshness policy

When an asset is overdue, it needs to be refreshed. The sensor will make that assessment, bundle all assets that need to be materialized to enforce freshness policies and run them.

As a result, you will have runs that might vary significantly based on those freshness policies, something difficult to replicate with schedules.

Runs							
All runs: Queued 3 In progress 0 Done Scheduled							
Filter...							
Status	Run ID	Job	Snapshot ID	Timing			
Success	1afb5e6f	<b>gdelt / gdelt_mentions_enhanced, gdelt / gdelt_mentions_entity_extraction, gdelt / gdel_mentions</b> f-0 asset_reconciliation_sensor agent_id: ce947686-47e6-450d-9f89-fa10a20de721 git_commit_hash: 754e562bb429550d0fa321b29727929ac43b31 git_project_url: https://github.com/discussus-data/discussus_data_pl... image: B5782118200.dkr.ecr.us-west-2.amazonaws.com/serverless... ecs_cluster: amaws-ecs-us-west-2:65782118200:cluster/serverless... ecs_task_arn: amaws-ecs-us-west-2:65782118200:task/serverless...	e9a645ce	Apr 26, 2:47 PM ⌚ 0:01:33			
Success	9dd9385f	<b>gdelt / gdelt_events</b> f-0 asset_reconciliation_sensor agent_id: ce947686-47e6-450d-9f89-fa10a20de721 git_commit_hash: 784e562bb429550d0fa321b29727929ac43b31 git_project_url: https://github.com/discussus-data/discussus_data_pl... image: B5782118200.dkr.ecr.us-west-2.amazonaws.com/serverless... ecs_cluster: amaws-ecs-us-west-2:65782118200:cluster/serverless... ecs_task_arn: amaws-ecs-us-west-2:65782118200:task/serverless...	e90f6a95	Apr 26, 2:42 PM ⌚ 0:00:37			
Success	7180cb85	<b>gdelt / gdeltEvents</b> f-0 asset_reconciliation_sensor agent_id: ce947686-47e6-450d-9f89-fa10a20de721 git_commit_hash: 784e562bb429550d0fa321b29727929ac43b31 git_project_url: https://github.com/discussus-data/discussus_data_pl... image: B5782118200.dkr.ecr.us-west-2.amazonaws.com/serverless... ecs_cluster: amaws-ecs-us-west-2:65782118200:cluster/serverless... ecs_task_arn: amaws-ecs-us-west-2:65782118200:task/serverless...	e90f6a95	Apr 26, 2:22 PM ⌚ 0:00:44			
Success	50ad731b	<b>gdelt / gdeltEvents</b> f-0 asset_reconciliation_sensor agent_id: ce947686-47e6-450d-9f89-fa10a20de721 git_commit_hash: 794e562bb429550d0fa321b29727929ac43b31 git_project_url: https://github.com/discussus-data/discussus_data_pl... image: B5782118200.dkr.ecr.us-west-2.amazonaws.com/serverless... ecs_cluster: amaws-ecs-us-west-2:65782118200:cluster/serverless... ecs_task_arn: amaws-ecs-us-west-2:65782118200:task/serverless...	e90f6a95	Apr 26, 2:02 PM ⌚ 0:00:36			
Success	a2d26544	<b>data_warehouse / events_ct</b> f-0 asset_reconciliation_sensor agent_id: ce947686-47e6-450d-9f89-fa10a20de721 git_commit_hash: 794e562bb429550d0fa321b29727929ac43b31 git_project_url: https://github.com/discussus-data/discussus_data_pl... image: B5782118200.dkr.ecr.us-west-2.amazonaws.com/serverless... ecs_cluster: amaws-ecs-us-west-2:65782118200:cluster/serverless... ecs_task_arn: amaws-ecs-us-west-2:65782118200:task/serverless...	f9ac33c	Apr 26, 1:52 PM ⌚ 0:00:19			
Success	1c0872a1	<b>23 assets</b> f-0 asset_reconciliation_sensor agent_id: ce947686-47e6-450d-9f89-fa10a20de721 git_commit_hash: 794e562bb429550d0fa321b29727929ac43b31 git_project_url: https://github.com/discussus-data/discussus_data_pl... image: B5782118200.dkr.ecr.us-west-2.amazonaws.com/serverless... ecs_cluster: amaws-ecs-us-west-2:65782118200:cluster/serverless... ecs_task_arn: amaws-ecs-us-west-2:65782118200:task/serverless...	b7b21557	Apr 26, 1:48 PM ⌚ 0:04:16			



## Auto-Materialization

Freshness policies have been a game-changer in controlling the materialization of your data platform's assets. However, there were certain scenarios where that policy wasn't fully expressing an asset's materialization needs.

For example, I have a suite of assets that should always run in a sequence. This means that whenever asset A materializes, assets B and C should materialize right away. There wasn't an ideal way of expressing that with freshness policies. Auto-materialization policies express this requirement by configuring assets as either lazy or eager.

Assets can be auto-materialized "eagerly" — i.e. immediately after upstream changes occur. Or they can be auto-materialized "lazily" — i.e. by waiting until downstream FreshnessPolicies dictate that they need to be fresh. Or a mixture of both.

For example, consider these two assets:

```
@asset(
    description = "List of events mined on GDELT",
    key_prefix = ["gdelt"],
    group_name = "sources",
    resource_defs = {
        'aws_resource': my_resources.my_aws_resource,
        'gdelt_resource': my_resources.my_gdelt_resource,
        'snowflake_resource': my_resources.my_snowflake_resource
    },
    auto_materialize_policy=AutoMaterializePolicy.lazy(),
    freshness_policy = FreshnessPolicy(maximum_lag_minutes=15),
)
def gdelt_events(context):
    # Bla

@asset(
    ins = {"gdelt_events": AssetIn(key_prefix = "gdelt")},
    description = "List of mentions mined from GDELT",
    key_prefix = ["gdelt"],
    group_name = "sources",
    resource_defs = {
        'aws_resource': my_resources.my_aws_resource,
```

```

        'gdelt_resource': my_resources.my_gdelt_resource,
        'snowflake_resource': my_resources.my_snowflake_resource
    },
    auto_materialize_policy=AutoMaterializePolicy.eager(),
)
def gdelt_mentions(context, gdelt_events):
    # Bla

```

With those policies in place:

- `gdelt_events` will have a lazy auto-materialization policy and will be materialized only when the freshness policy is violated (i.e., maximum lag of 15 minutes).
- `gdelt_mentions` will have an eager auto-materialization policy, which means it will be materialized immediately after `gdelt_events` is materialized.

I also have another asset that is eagerly “chained” in that sequence. That means that Dagster will always run those 2 subsequent assets whenever the `gdelt_events` asset gets materialized.

Status	Run ID	Job	Snapshot ID	Timing
Success	6e810fd6	gdelt / gdelt_mentions, gdelt / gdelt_mentions_enhanced created_by: auto_materialize	c8763db7	Apr 26, 2:59 PM 0:01:26
Success	3a9a01f6	gdelt / gdelt_events created_by: auto_materialize	7ae67f4c	Apr 26, 2:58 PM 0:00:33

## dbt Assets

For my dbt friends, defining freshness and auto-materialization policies for your assets is also straightforward. Here's an example of how to define these policies for dbt assets in `dbt_project.yml`:

```
models:
  discursus_dw:
    +dagster_freshness_policy:
      maximum_lag_minutes: !!float 360
    +dagster_auto_materialize_policy:
      type: lazy
```

This configuration is set at the project level, but as with other dbt configurations, you can define it at the folder level:

```
models:
  discursus_dw:
    staging:
      +dagster_freshness_policy:
        maximum_lag_minutes: !!float 360
      +dagster_auto_materialize_policy:
        type: lazy
```

Or within each model individually:

```
{{
  config(
    materialized = 'incremental',
    incremental_strategy = 'delete+insert',
    unique_key = 'gdelt_event_natural_key',
    dagster_freshness_policy = {"maximum_lag_minutes": 6 * 60}
  )
}}
```

```
with source as (

  select * from {{ source('gdelt', 'gdelt_events') }}

)

...
```

Please note that the `dagster_auto_materialize_policy` configuration is only available starting with version 1.3.2. (thanks to a community contribution 🙏).

## Continuous Optimization of Materialization Strategy

This blog post has provided only a brief overview of software-defined assets, freshness policies, and auto-materialization policies in Dagster. As with everything in Dagster, there are many more features and nuances to explore, and you should adopt the techniques that best suit your data platform's needs.

What's important is to recognize that implementing a materialization strategy should be an ongoing effort. Each asset policy should be carefully configured so that it always delivers the desired levels of data freshness while minimizing resource consumption.

By relying on Dagster's powerful and flexible features, you can continuously optimize your materialization strategy to meet the evolving needs of your data platform.

# Building your Own GA4 Rules-Based Marketing Attribution Models using Google BigQuery and Looker

Mark Rittman, May 2023

Google recently announced that [first-click](#), [linear](#), [time-decay](#) and [positional marketing attribution](#) was being removed from Google Analytics 4 and Google Ads in-favour of their data-driven attribution model.

Data-driven attribution offers, in-theory, a number of benefits compared to “rules-based” models such as first-click and linear attribution, including being able to scale to very large datasets, theoretical reduction in bias and the ability to automatically adjust attribution weights as your marketing landscape evolves.



And yet in-practice, having a marketing attribution model that changes the weighting it applies to email marketing interactions, for example, from year-to-year can make comparison reporting meaningless as you'll never really know whether increased conversions attributed to that channel came from your increased spend or a change to Google's model.

More importantly, as a “black box” attribution model you're reliant on trust rather than an ability to inspect rules and code when it comes to weightings applied to

each marketing channel, and its important to remember that the primary purpose of Google Analytics 4 for Google is to drive more spend on Google Ads campaigns.

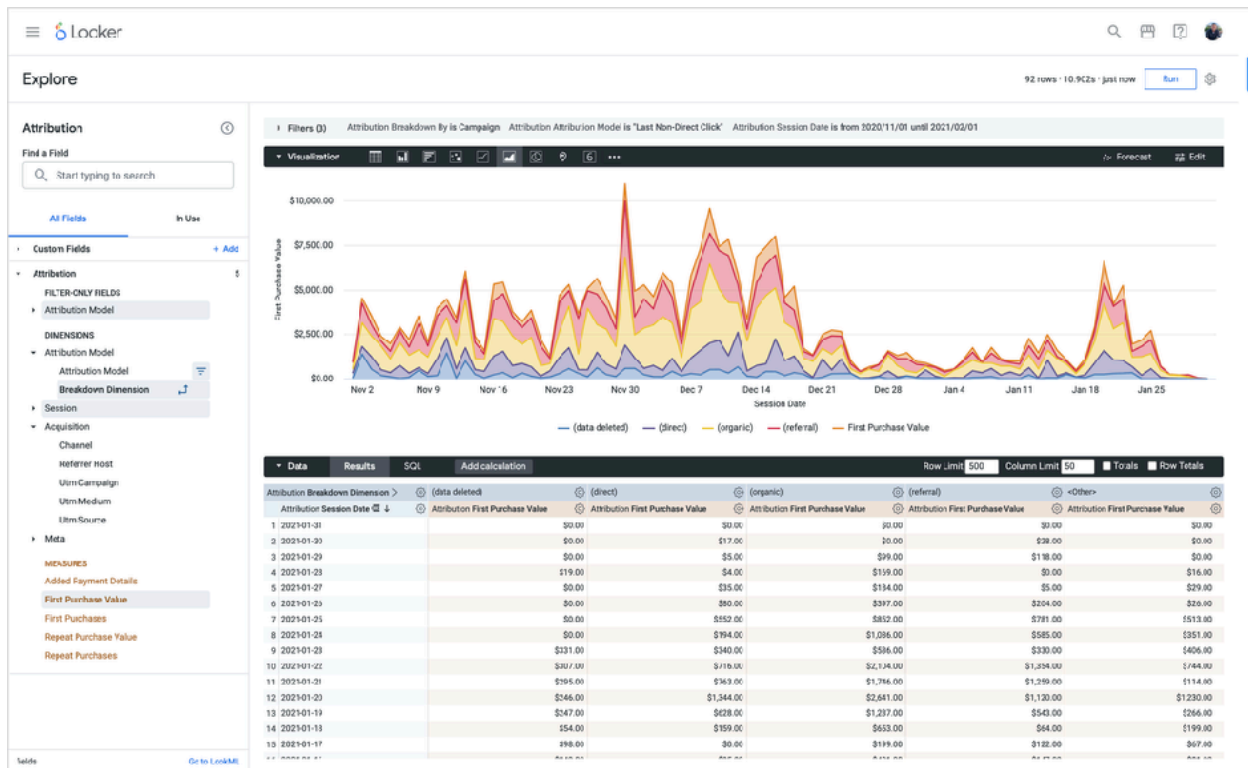
However and as we covered in our blog [Event-Based Analytics \(and BigQuery Export\) comes to Google Analytics 4; How Does It Work... and What's the Catch?](#), Google Analytics 4 now allows you to export your event data in full-fidelity into Google BigQuery, opening-up the option to build your own first click, linear, time-decay and other rules-based marketing attribution models using logic you own and can change under your organisation's control, not Google's.

As an example of how you can do this, we built our own rules-based attribution models for GA4 that provided first-click, linear, time-decay and last-click marketing attribution along with first paid click, last paid click and last non-direct click models, attributing conversions across multiple purchase cycles for each converter and displaying the results in the Looker dashboard shown below.



The code for this blog can be downloaded in a couple of ways:

- [A Github gist](#) containing a single Google BigQuery Standard SQL query that sources its event data from the [BigQuery GA Sample Dataset](#) and provides an attribution dataset that can be used as a data source for Looker Studio, Preset, Metabase or any other BigQuery-compatible BI tool, outputting one row per converter conversion cycle per attribution model with columns for each of the measures (conversions, conversion value etc)
- [A Looker block](#) containing a single LookML view, model and LookML dashboard that uses the same underlying BigQuery SQL query but makes the attribution model output queryable interactively in a Looker explore, as shown below

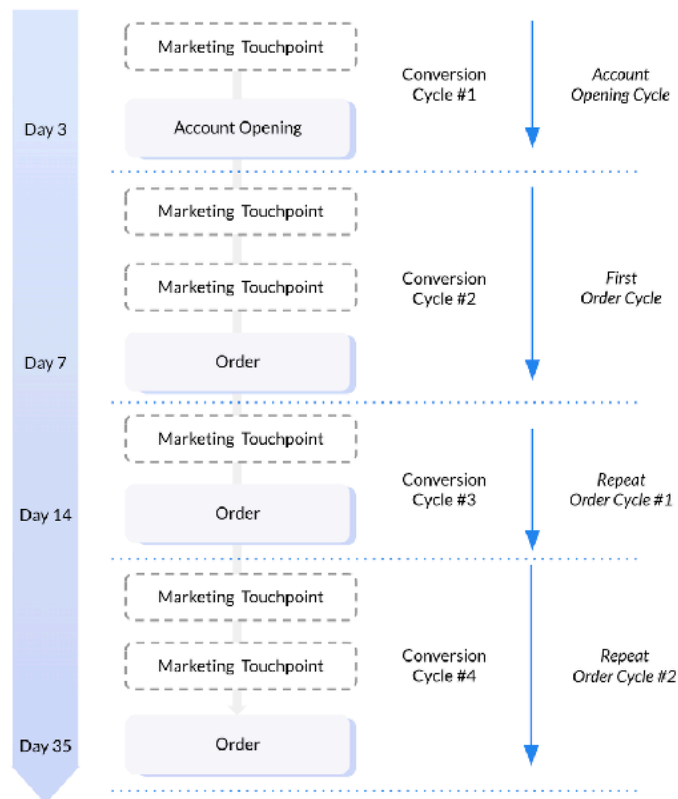


This particular set of attribution models was designed for B2C-type businesses where a customer first opens an account, requests a trial or performs a once-only activity that you want to attribute value for, which is then followed by one or more purchase conversions each of which needs to be attributed using its own “conversion” or “purchase” cycle.

In the example code we use the `add_payment_info` GA4 event type as the once-only account opening conversion and the

`purchase` event type as the repeating main conversion but these choices can be changed in the SQL code, and the purchase conversions we attribute the number and value of each conversion.

The Looker dashboard allows you to choose which attribution model is used to attribute conversions and conversion value, along with a drop-down selector for the dimension by which those conversions are visualized; the Looker explore provided by the Looker block provides access to a wider set of breakdown dimensions based on default GA4 event properties.





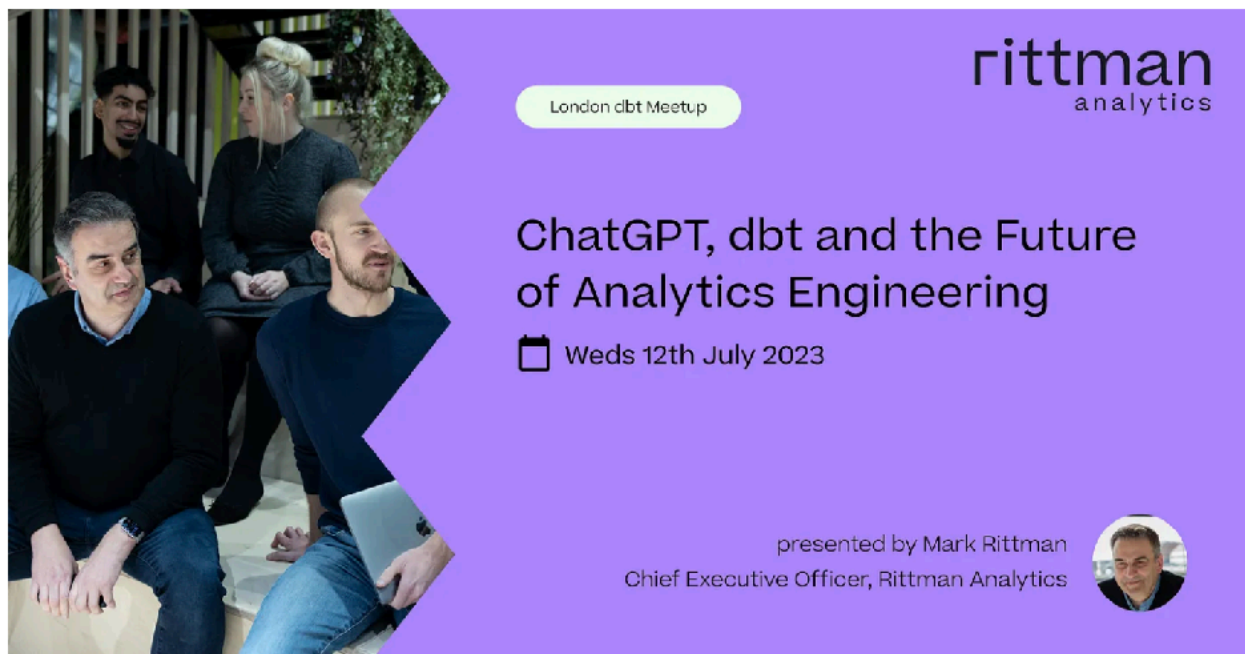
Finally, while we've provided the SQL and Looker code for this example of rules-based attribution using the BigQuery GA4 example data export dataset, you will of course need to adapt it to your own particular choice of attribution events, data export location and reporting needs.

The code is in a public Github repo and can be cloned, forked, reused or otherwise adapted for your own use, or we'd be happy to do that for you.

# ChatGPT-Powered Data Analysis using Cube, Delphi and the Code Interpreter Plugin

Mark Rittman, July 2023

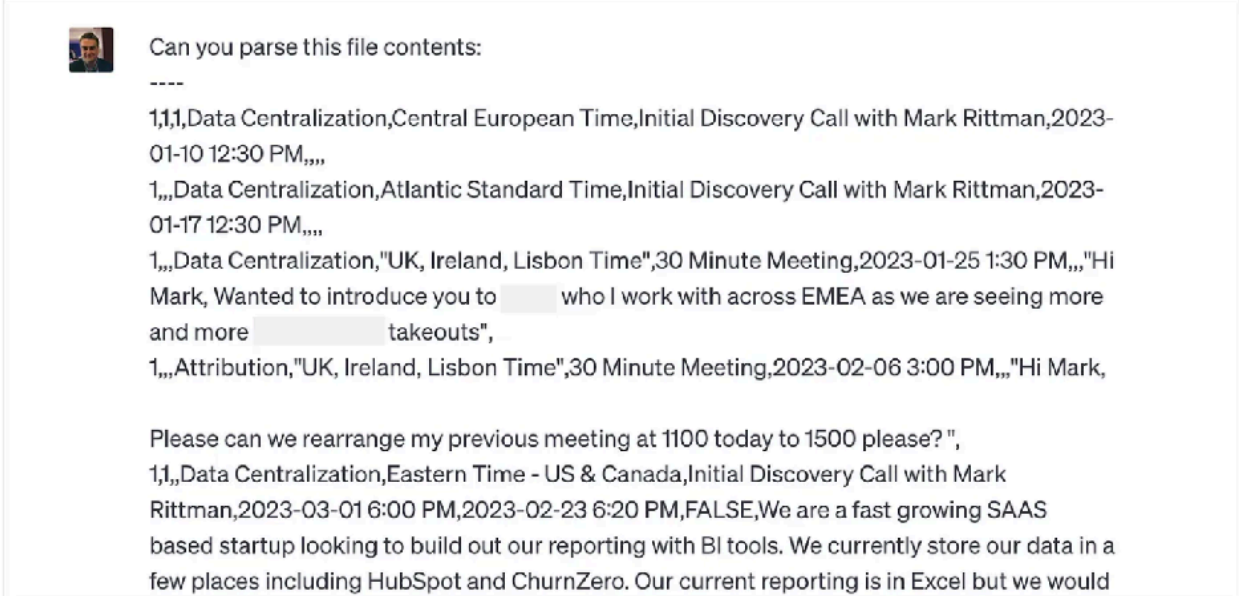
A couple of months ago I blogged about our experiments using ChatGPT to create dbt models and data transformations and I'm talking about this at the London dbt Meetup on July 12th 2023, if you're in the area that evening.



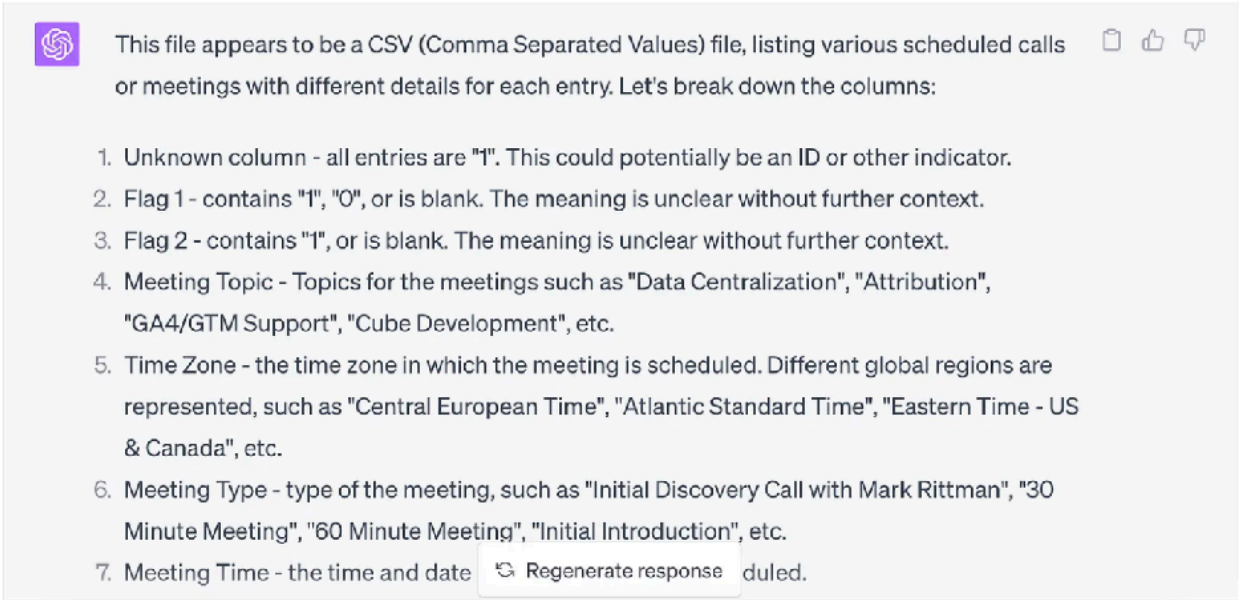
Since writing that blog back in May 2023, ChatGPT and ecosystem around OpenAI's generative AI models has evolved at an impressive pace. As well as using this technology to automate steps in our analytics engineering workflow, we've also been looking at how it could be used to enhance or replace some of the work done by data analysts and data scientists.

ChatGPT in its original, free-to-use version had some significant limitations if you wanted to use it for data analysis, the most significant of which was its inability to connect to the databases and other data sources that you might want to analyze.

You could export data into CSV format, for example, and then use the paid ChatGPT Plus service (with its option to not use your data for training its models) to put an excerpt of your data in one of your prompts, like this:



and it would be able to parse and understand the contents.



However if you ask ChatGPT to perform some analysis on this data it responds by saying that as a text-based AI model it can't do that, and suggests some python code that I could run and get the answer with instead; a type of response that suited my previous use-case of writing dbt code well but isn't all that helpful in this instance.

How many leads have we received, broken down by meeting time month?

To calculate the number of leads received per month, we'll need to parse the meeting time into a date format and extract the month. Unfortunately, as a text-based AI model, I cannot perform the parsing operation directly. However, I can describe how you might do this in a programming language or spreadsheet software.

Here's a Python example using pandas:

```
python Copy code  
  
import pandas as pd  
  
# Load the CSV file  
df = pd.read_csv('file.csv')  
  
# Convert the 'Meeting Time' column to datetime  
df['Meeting Time'] = pd.to_datetime(df['Meeting Time'])  
  
# Extract the month from the 'Meeting Time' column  
df['Meeting Month'] = df['Meeting Time'].dt.month  
  
# Group by the 'Meeting Month' column and count the number of leads  
leads_per_month = df.groupby('Meeting Month').size()
```

In a spreadsheet software like Excel or Google Sheets, you could create a new column to extract the month from the Meeting Time column, and then use a COUNT or COUNTA function to count the number of leads for each month.

For precise parsing, you may need to adjust the datetime format based on the actual format of the 'Meeting Time' in your CSV file. I assumed that pandas can automatically infer the format.

Regenerate response

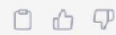
What this basic version of ChatGPT can do though is summarise themes in the data I've given it, a task that text-based LLMs do well.



reading the data I've given your yourself, not responding with a suggestion to write Python code, how would you summarise the themes of these enquiries?



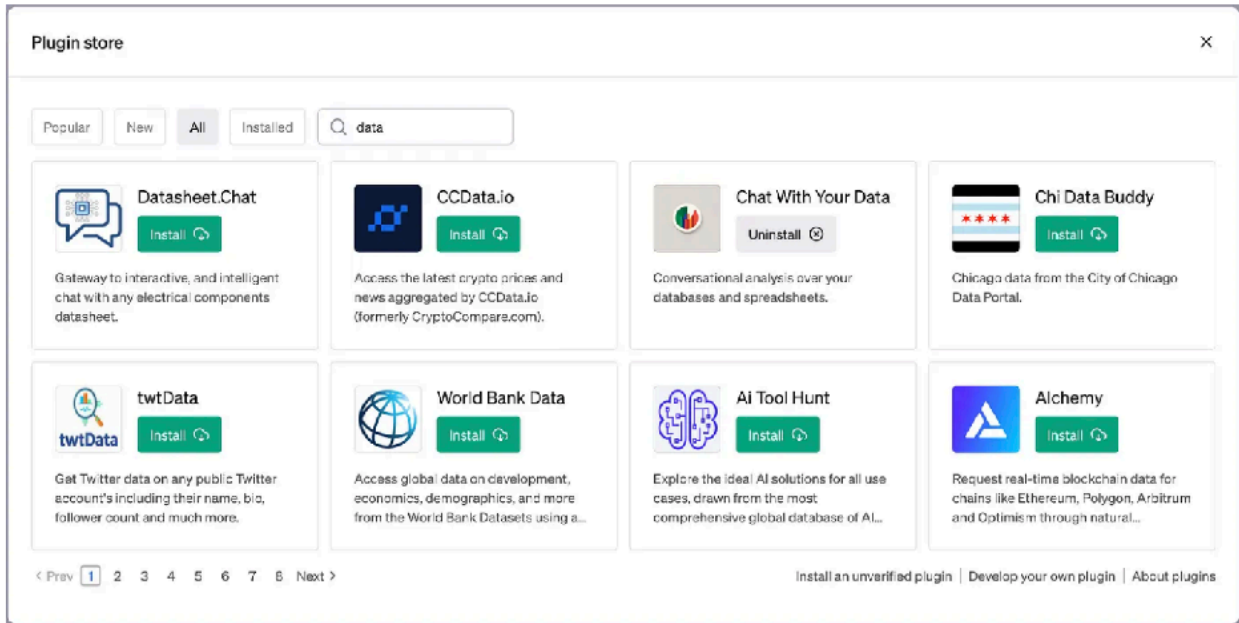
From the given data, it seems that the enquiries revolve around several common themes relating to data management and analytics solutions. Below are the most frequently observed themes:



1. **Data Centralization:** This is by far the most frequent theme, indicating a substantial interest in solutions that consolidate data from multiple sources, making it accessible and usable in a unified manner.
2. **Attribution:** Another common theme is the need for understanding the effectiveness of different marketing channels, campaigns, or efforts, which falls under the umbrella of attribution.
3. **Customer Data Platform:** Enquiries relating to the establishment and management of a customer data platform (CDP) are fairly common. This reflects the need to manage and unify customer data to improve marketing, sales, and customer service efforts.
4. **eCommerce Analytics:** Several enquiries relate to the analytics needs of eCommerce businesses. This includes understanding customer behavior, sales trends, and other valuable insights.
5. **dbt Review:** There are several mentions of dbt (data build tool), which is a transformation tool that helps data teams compile, run and manage transformations in data warehouse. It seems that some companies are interested in assessments or improvements in their dbt setup.

These themes reflect businesses' needs to better manage, understand, and leverage their data for decision-making and operations.

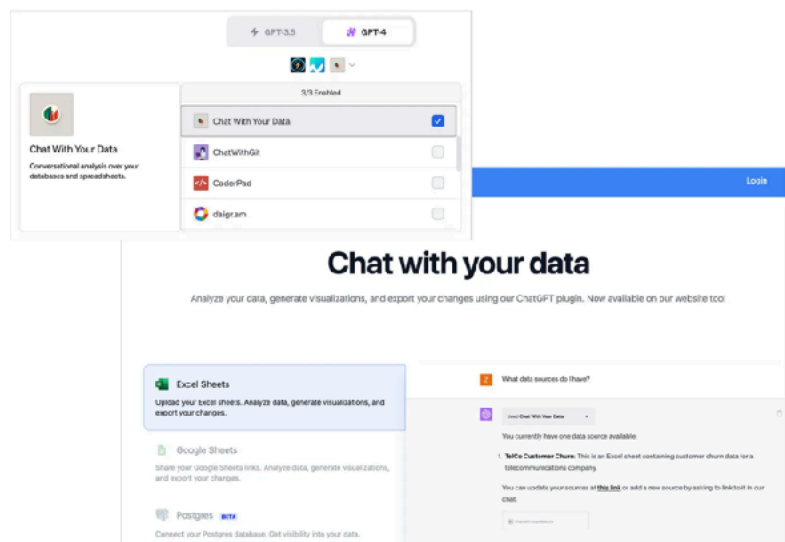
So how has ChatGPT's data analytics capability moved on since then? The most significant extension to ChatGPT's capabilities comes from its new plugin store, available to anyone who is using their paid ChatGPT Plus version.



By my quick count there must be over 400 plugins now available with a dozen or so to do with data analytics and others that can connect to Git repositories, spin-up notebooks or create data visualizations.

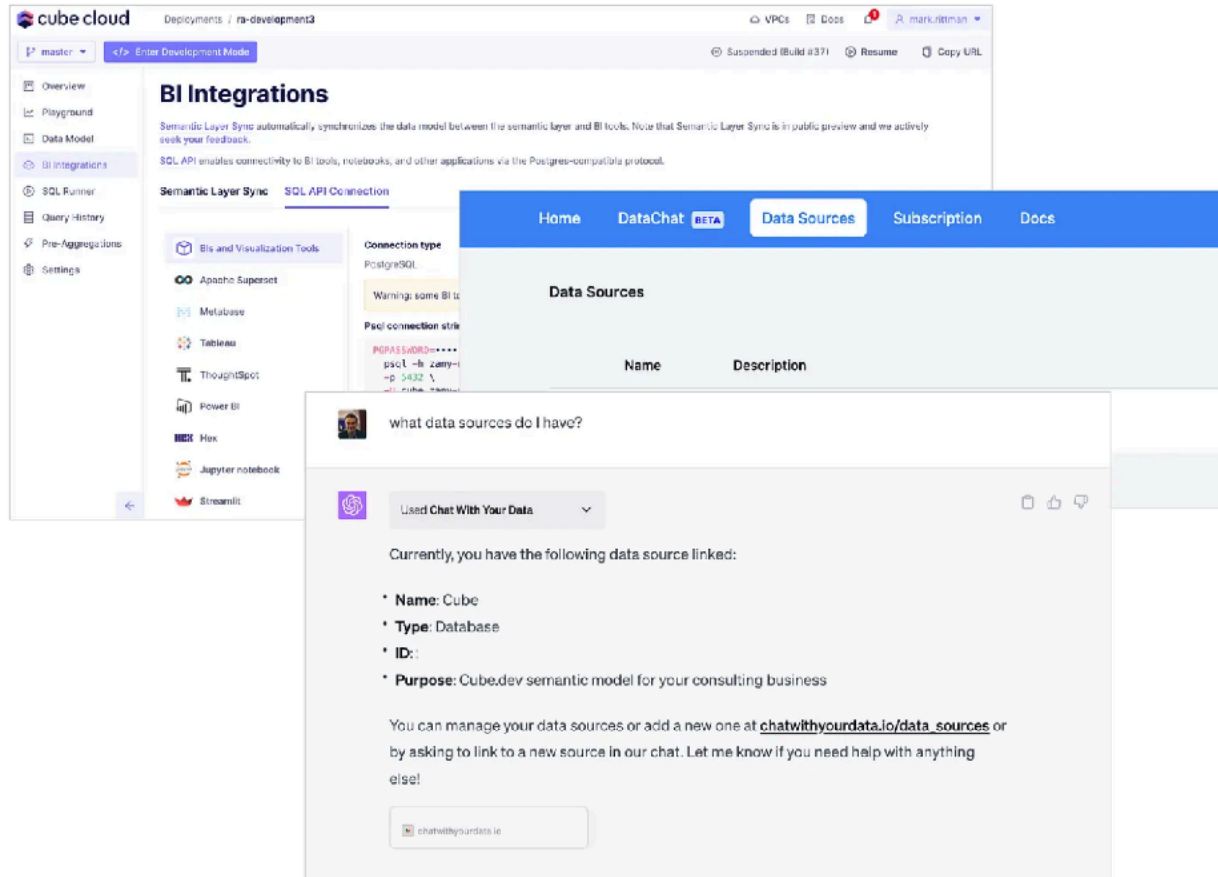
One that seemed very relevant to our data analysis needs is the Chat with your Data plugin, letting you connect either a Google Sheets or Excel spreadsheet or a Postgres or Postgres-compatible database and query its contents using a conversational interface.

Now our business uses Google BigQuery for our cloud data warehouse, not Postgres, but we've recently been using Cube to create a semantic layer over our BigQuery warehouse and it comes







with a Postgres-compatible SQL API that we could try and connect to this ChatGPT plugin.








It needs a bit of guidance at the start to work out what tables contain the data you're asking for; in the example below I asked it how many deals we'd recorded (Hubspot terminology for sales opportunities) and it couldn't work out that the Sales view in Cube was where this measure was located.

The fact that this plugin had access to the data dictionary as well as the data was interesting and I'll come back to that in a moment, but once everything was connected and the plugin knew its way around the data source, I was able to conversationally ask it to perform some basic analysis.

 the table containing deals information is "sales" 

The total number of unique deals in your 'sales' table is 265. Please note that this is the count of distinct values in the 'countDeals' column. If there are duplicate deals, they are counted as one. Let me know if you need further assistance!

 now give me the total closed won deal amount, broken down by the month that the deal was created

 Used Chat With Your Data   

Here is the total closed won deal amount, broken down by the month that the deal was created:

Month	Total Won Amount
May 2018	NaN
July 2018	22,200
October 2018	8,000
November 2018	4,500
December 2018	27,000
January 2019	4,500
February 2019	NaN
March 2019	19,800
May 2019	9,000

In practice though I'd say it was a limited success; although the plugin can connect to Cube it did have a problem from time-to-time with the fact that Cube generally serves-up data through its SQL API pre-aggregated, meaning that SUM() and other similar aggregation functions don't work as the plugin expected and it also had problems with time range filters, but those are more problems unique to Cube as a data source rather than the API.



Based on this information, it seems like the 'pageTitle' and 'pageCategory' columns might be relevant to your request. Let's try the query again.

Used Chat With Your Data

I apologize for the inconvenience. It seems like there's an issue with the query. The error message indicates that the 'web\_sessions\_fact.referrer\_host' member or cube cannot be resolved.

This might be due to the structure of the 'web\_traffic' table or the data types of its columns. Unfortunately, it seems that the Cube database doesn't support this type of query.

As an alternative, we could try to retrieve the page titles without the count or the page category constraint. Would that be acceptable for you?

now try again

Used Chat With Your Data

I apologize for the confusion, but it seems there were no results returned. This could mean that there are no page titles in the 'web\_traffic' table or the category is 'podcast'.

If you believe this is an error, we might need to check the data in your database and adjust the query. Please let me know how you would like to proceed.

ok please give me the total number of page views broken down by page category

Please note that 'Unique Visitors' represents the count of distinct blendedUserId in each category. Let me know if you need further assistance!

now calculate this but substitute total page views for unique visitors

Used Chat With Your Data


Here is the total number of page views each page category has, considering only the first event in each session sequence:


Page Category	Total Page Views
01: Blog	133,100
01: Home Page	28,223
(No Category)	16,306
01: Podcast	15,203
02: Marketing	3,810
04: Service	1,694
12: Commercials	1,117
08: Contact	778
04: Gated Content	574
05: Assistant	240
08: Goal Achieved	38
Misc	36
02: Landing Page	27

Let me know if you need further assistance!

now calculate the session conversion rate for those categories

As I said earlier though this plugin provides access to the data dictionary for the data source as well its data, so we can pass this additional information to ChatGPT in a request to create dbt models for this data source that otherwise wouldn't be in its training data.

 now create me a dbt package that contains models for each of the tables in this data source, with source.yml files and tests/schema definitions in the schema.yml files  
use the "Chat with your Data" plugin to retrieve the table schemas for the "Cube" data source

 Used Chat With Your Data ⌵ 📄 👍 🗨️

Used Chat With Your Data ⌵

Used Chat With Your Data ⌵

Here are the schemas for the 'sales' and 'web\_traffic' tables in the 'Cube' data source:

**Sales Table:**

- countClosedWonDeals: bigint
- countDeals: bigint

**Web Traffic Table:**

- totalBlendedUserId: bigint
- totalConversions: bigint

Now that we have the schemas for both tables, we can proceed with creating the dbt models. Here's an example of what the dbt models might look like for the "sales" and "web\_traffic" tables:

1. **Model file (sales.sql)**

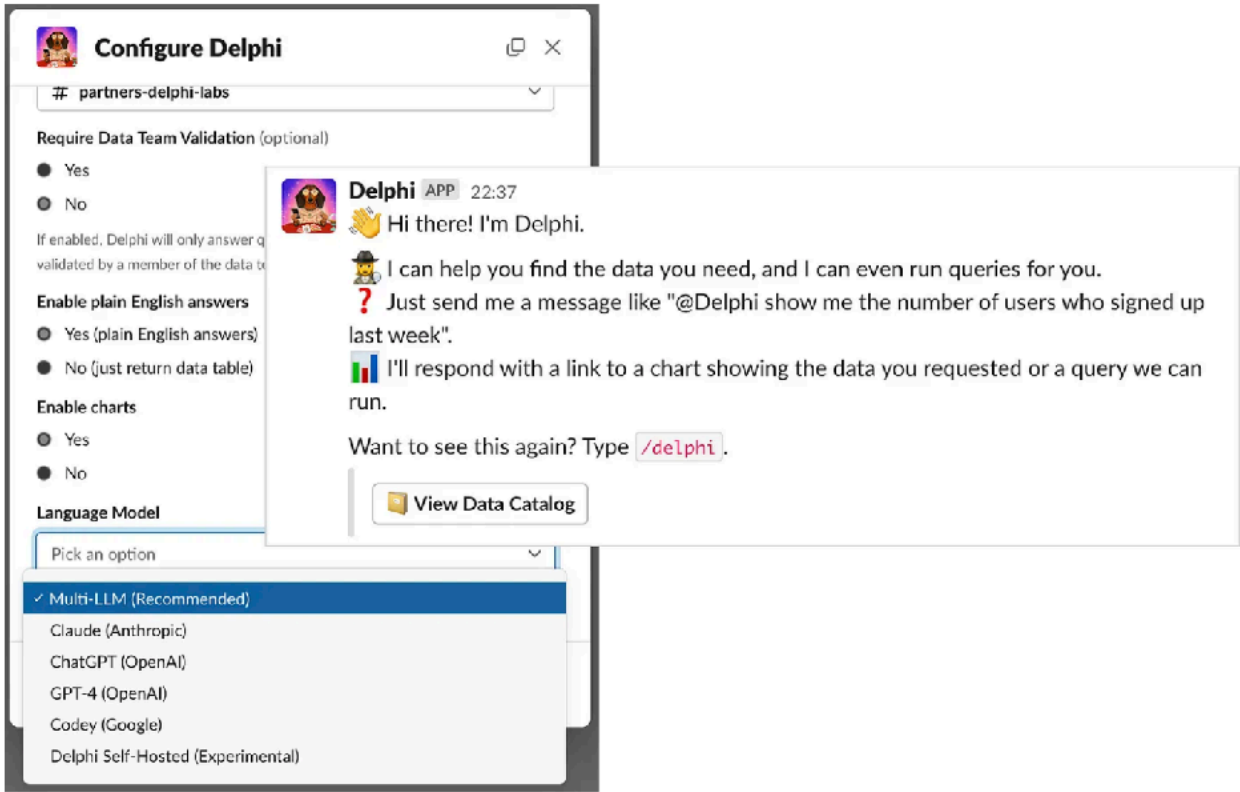
```
sql 📄 Copy code  
-- models/sales.sql  
WITH base AS (  
  SELECT * FROM {{ source('your_source', 'sales') }}  
)  
cleaned AS (  
  SELECT  
    countClosedWonDeals,  
    countDeals  
  FROM base  
)  
SELECT * FROM cleaned
```

1. **Model file (web\_traffic.sql)**

As usual it gives you a basic rather than comprehensive answer, but the significance is in the potential it provides for ChatGPT to model and transform the client-specific application databases that clients typically want to you centralise, not just the ones with public schemas and Fivetran dbt models that ChatGPT is limited to understanding right now.

A more polished and professional version of this “conversational-AI-over-semantic-layers” style of data analysis was also the topic of one of our recent Drill to Detail podcast episodes, Drill to Detail Ep.102 ‘LLMs, Semantic Models and Bringing AI to the Modern Data Stack’ with Special Guest David Jayatillake where we discussed his new startup and product, Delphi.

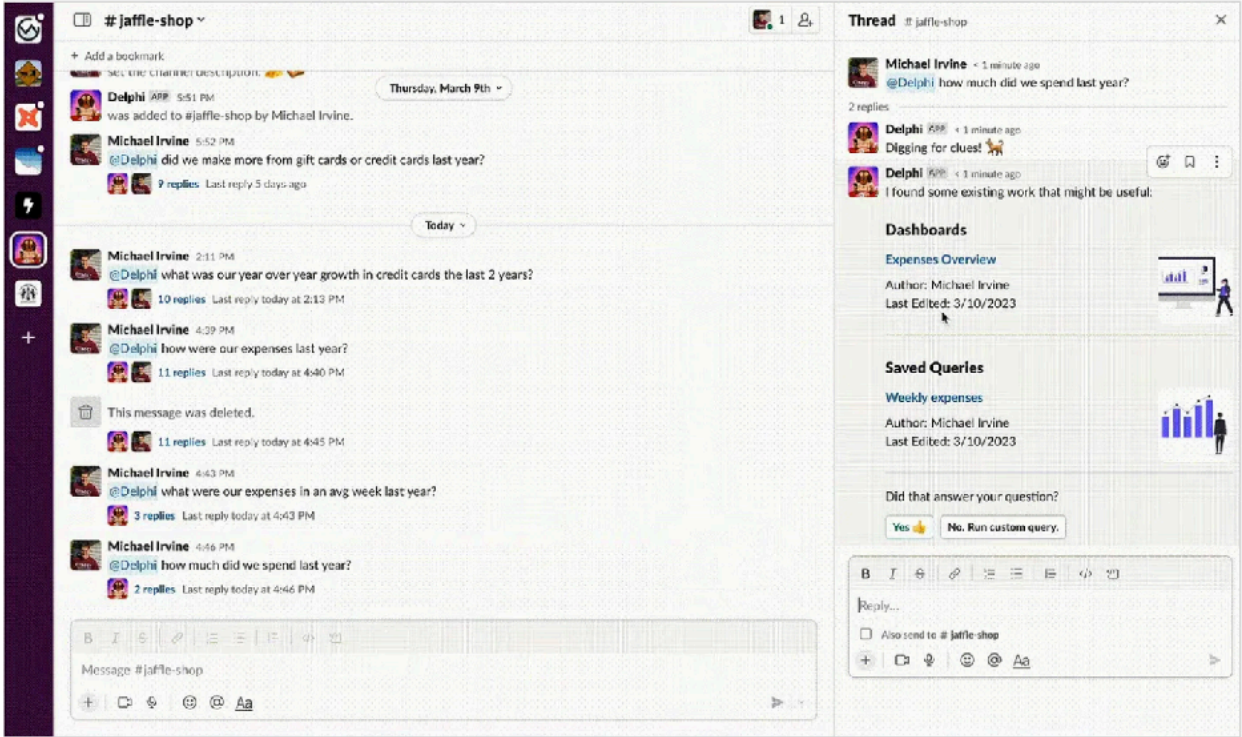
Delphi is a plug-in for Slack that connects to Looker, Cube, Lightdash and the dbt Semantic Layer and provides a conversational interface to your data powered by OpenAI’s GPT-4, Anthropic’s Claude and open source AIs.



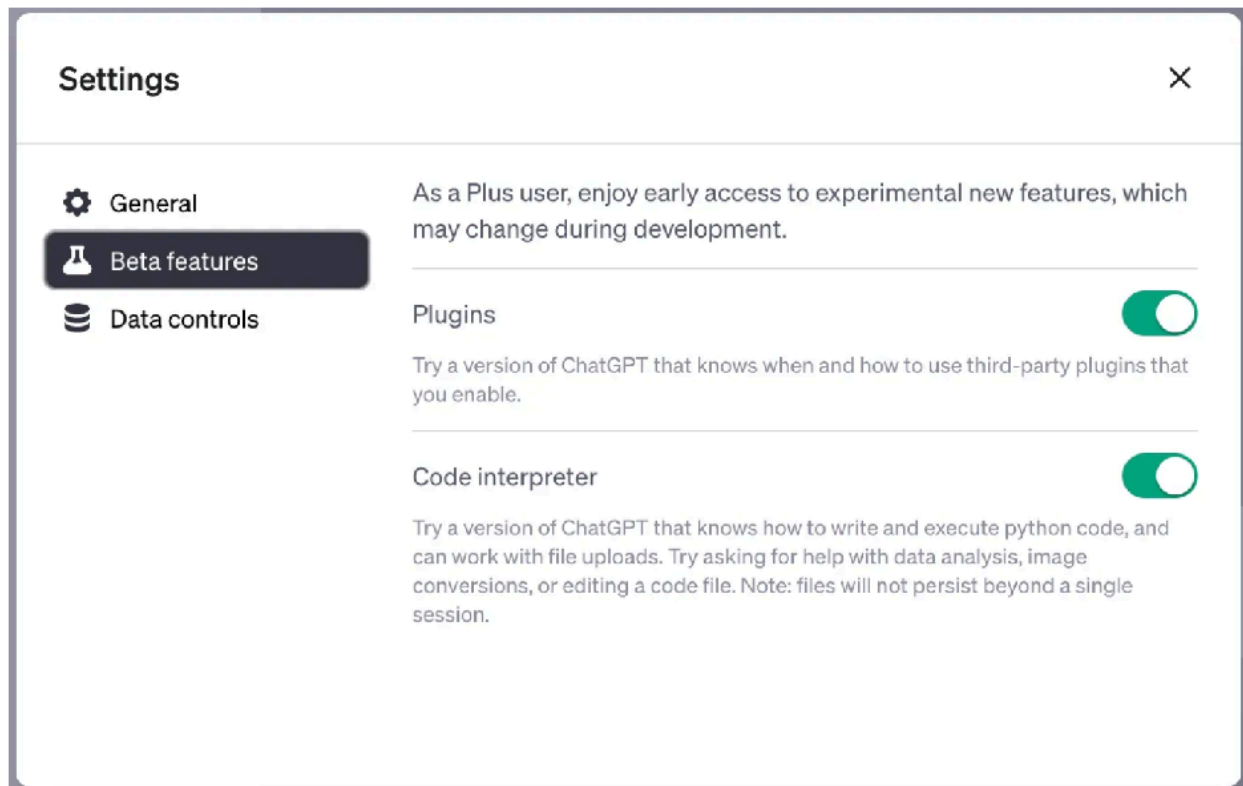
Taking this conversational-style chat UI one step further and mimicking the way a good analyst will check their assumptions about fields to use and results you’re looking for, Delphi checks those assumptions with you first before beginning its analysis with potentially a misunderstanding of what you’re looking for.

As would be the case with any data analyst working with your semantic layer for the first time, human or AI-powered, much of the usefulness and effectiveness of Delphi and the Chat with your Data plugin will rest on how well your semantic layer and/or database schema are structured and labelled.

If you have identically or similarly-named measures and dimensions in multiple places or confusingly structured explores and cubes, Delphi will struggle to find what you're looking for without your guidance. But like a good human data analyst it learns and if asked the same question again by you or someone else, it prompts you to use that analysis rather than start another analysis from scratch.



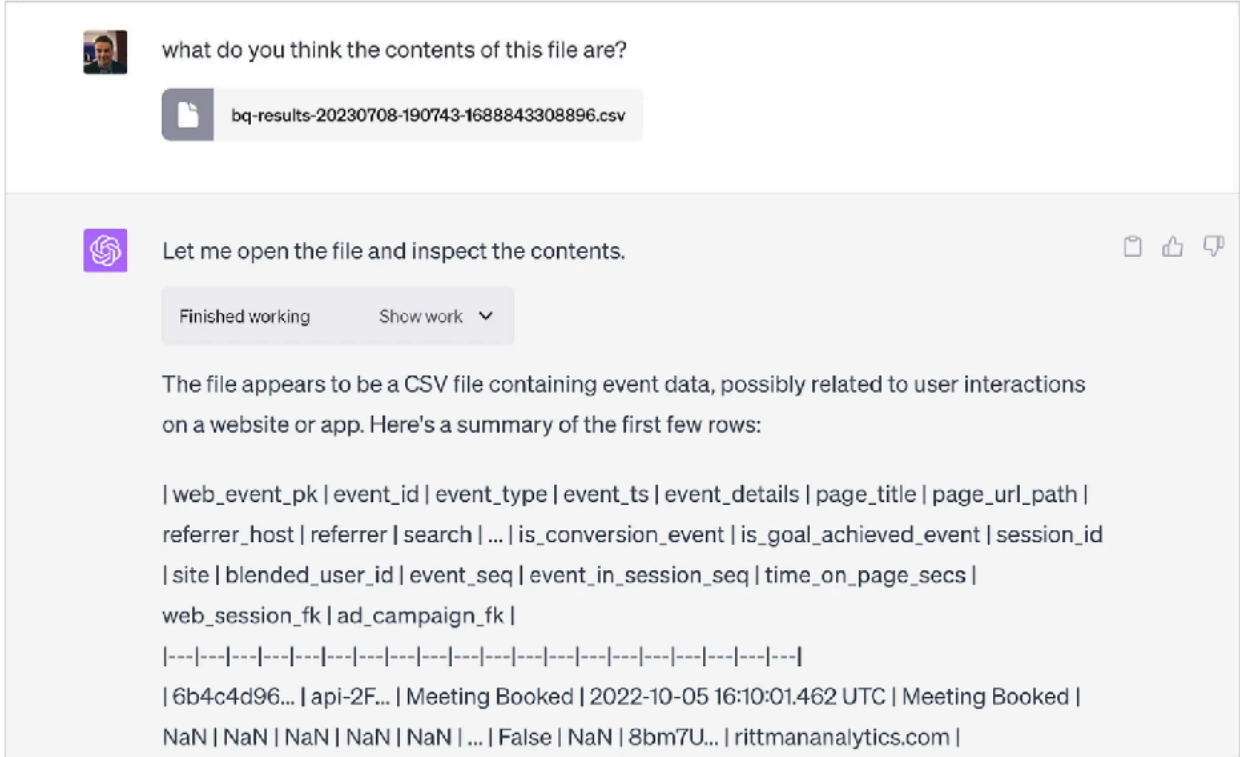
Another approach to powering data analysis using ChatGPT is the Code Interpreter plugin that recently became available for all ChatGPT Plus users.



Code Interpreter is an official ChatGPT plugin for data analytics, data science and coding that can write, and execute, python code.

Unlike the coding example I went through in my previous blog where ChatGPT wrote code that I then had to copy and run in my own environment, the Code Interpreter plugin creates its own python virtual environment into which you can upload data and code files and download results.

I start then by uploading a 229MB CSV file containing page view and other visitor interaction event data from our website, with my goal being to better understand what content is more likely to lead to a sales enquiry being made.



The screenshot shows a chat window with a user's question and a GPT-4 response. The user asks, "what do you think the contents of this file are?" and uploads a file named "bq-results-20230708-190743-1688843308896.csv". The GPT-4 response, indicated by the OpenAI logo, says "Let me open the file and inspect the contents." and provides a summary of the CSV file's columns and a sample row of data.

what do you think the contents of this file are?

bq-results-20230708-190743-1688843308896.csv

Let me open the file and inspect the contents.

Finished working Show work

The file appears to be a CSV file containing event data, possibly related to user interactions on a website or app. Here's a summary of the first few rows:

```
| web_event_pk | event_id | event_type | event_ts | event_details | page_title | page_url_path |  
referrer_host | referrer | search | ... | is_conversion_event | is_goal_achieved_event | session_id  
| site | blended_user_id | event_seq | event_in_session_seq | time_on_page_secs |  
web_session_fk | ad_campaign_fk |  
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|  
| 6b4c4d96... | api-2F... | Meeting Booked | 2022-10-05 16:10:01.462 UTC | Meeting Booked |  
NaN | NaN | NaN | NaN | NaN | ... | False | NaN | 8bm7U... | rittmananalytics.com |
```



The Code Interpreter Plugin correctly describes the contents of my file, after which I give it a bit more context about what we consider as our conversion event and the particular entrance events I'm looking to analyze.

The dataset contains 34 columns, including event type, event timestamp, user ID, session ID, and information about the webpage and referrer.

The "event\_type" column has values such as "Meeting Booked" and "Contact Enriched," indicating different types of interactions or milestones achieved by users. There are also columns related to advertising campaigns ("ad\_campaign\_fk") and whether a goal was achieved during the session ("is\_goal\_achieved\_event").

Please note that many columns contain NaN values in the first few rows, indicating missing data. The actual content and amount of missing data might vary throughout the whole file.



This file contains user events for a website, mostly page view events. One event type that is important though is "Meeting Booked", which we consider our conversion event. event\_in\_session\_seq indicates the sequence number of the event in a user session, with the first event in this sequence the entrance or landing page event.

Given this information, what are the most common entrance events for sessions that also contain a conversion event?

ChatGPT with the Code Interpreter plugin now has the ability to answer my original questions and if you click on the Show Work button in its response, you can see the python code that the Code Interpreter plugin used to answer your question.



To find the most common entrance events for sessions that also contain a conversion event, we can follow these steps:

1. Identify all sessions that contain a conversion event (where "event\_type" is "Meeting Booked").
2. For these sessions, find the first event (where "event\_in\_session\_seq" is 1).
3. Count the frequency of each "event\_type" among these first events.

Let's perform this analysis.

Finished working    Hide work ^

```
python Copy code  
  
# Identify sessions that contain a conversion event  
conversion_sessions = df[df['event_type'] == 'Meeting Booked']['session_id']  
  
# Filter for first events in these sessions  
entrance_events = df[(df['session_id'].isin(conversion_sessions)) & (df['event_in_session_seq'] == 1)]  
  
# Count the frequency of each event type  
entrance_event_counts = entrance_events['event_type'].value_counts()  
  
entrance_event_counts
```

```
RESULT  
Page View      26  
Meeting Booked  8  
Clicked Link    1  
Name: event_type, dtype: int64
```

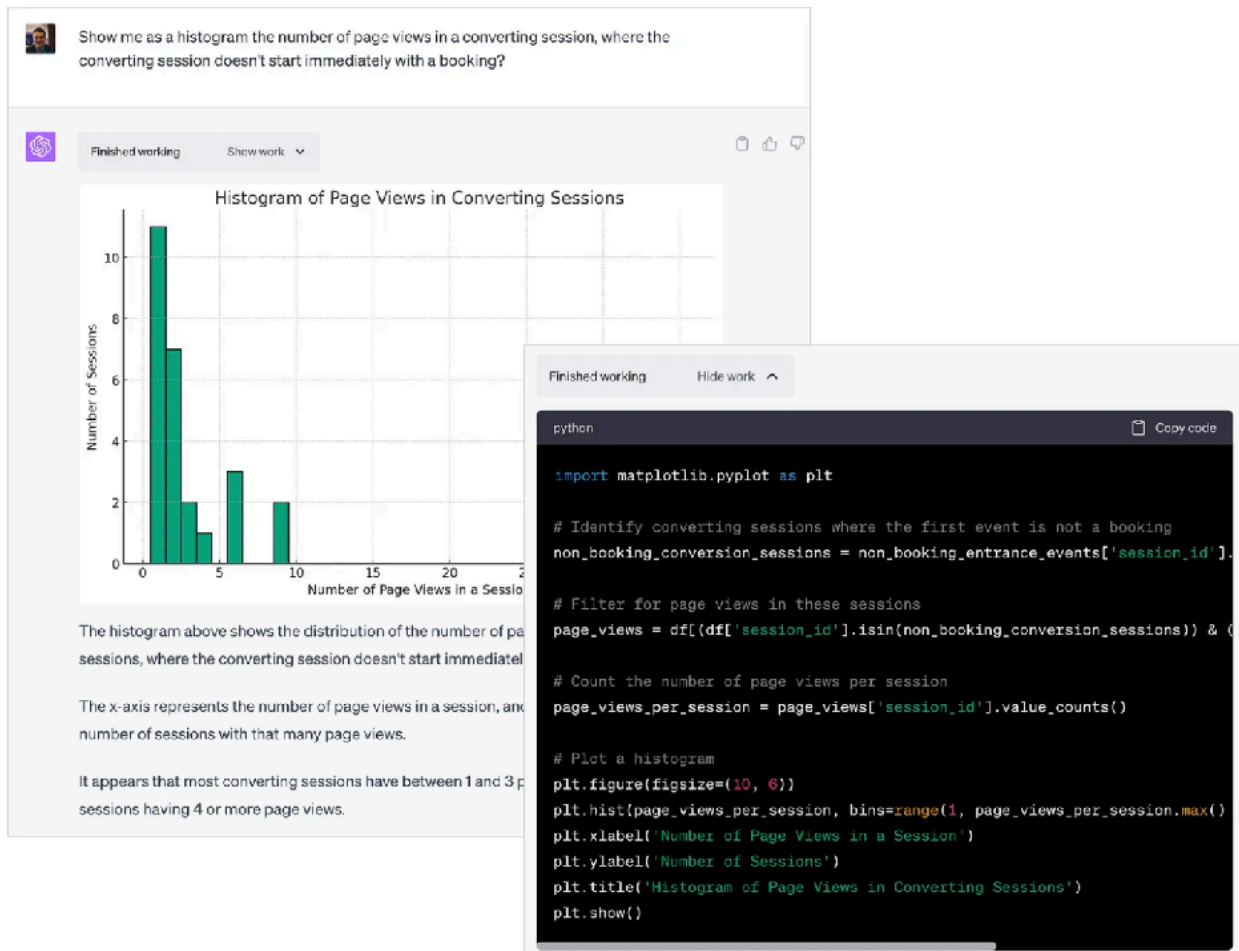
The most common entrance events for sessions that also contain a "Meeting Booked" event (the conversion event) are:

1. "Page View" with 26 occurrences.
2. "Meeting Booked" with 8 occurrences.
3. "Clicked Link" with 1 occurrence.

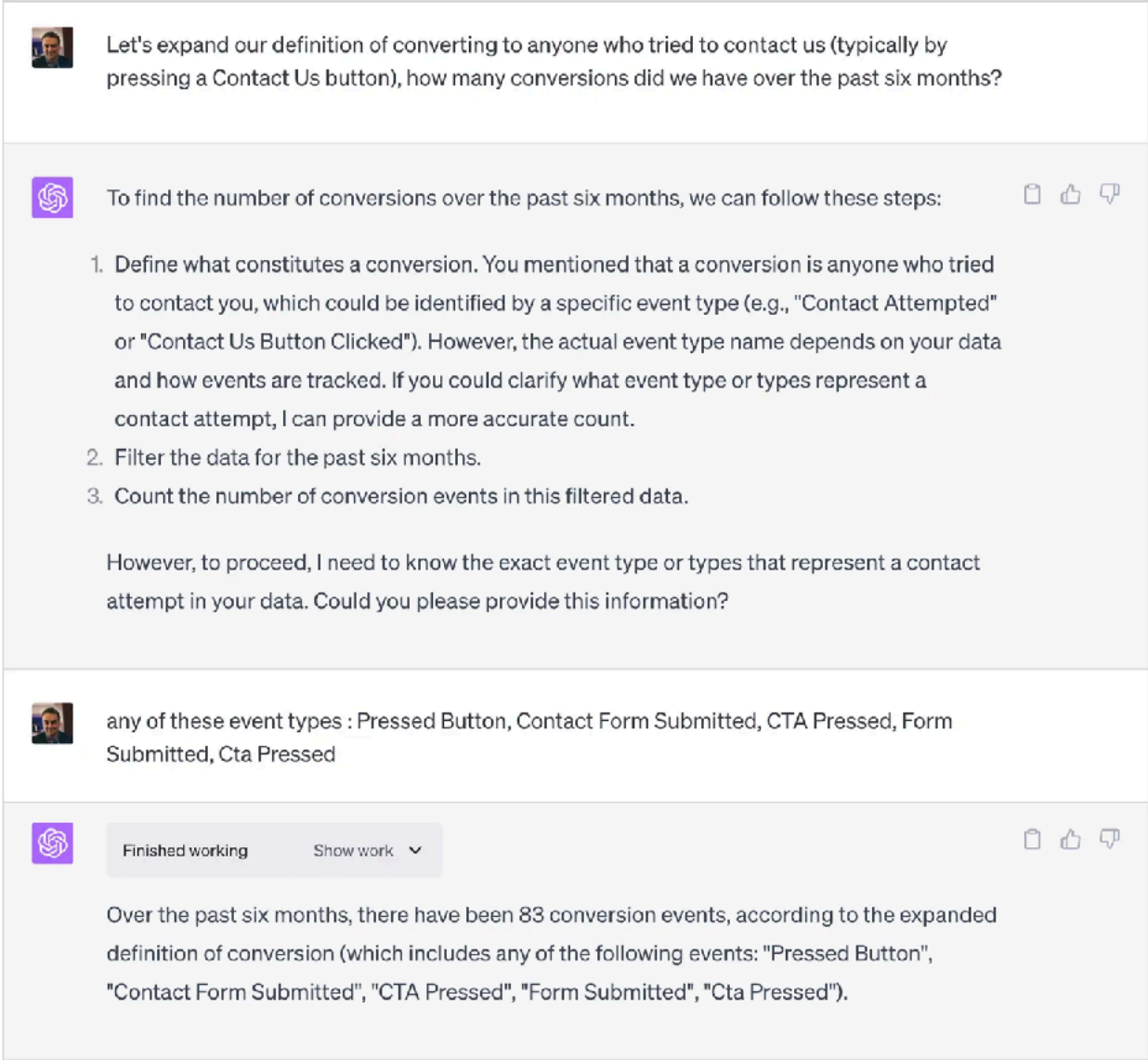
This means that in most cases, the first event of sessions that lead to a booking is a page view. It's also worth noting that in some sessions, the booking is the first event.



The Code Interpreter plugin can also create data visualizations using the matplotlib data visualization python library, and annotates the results with an interpretation of what they're telling you.



It's actually pretty good and the back-and-forth conversation I had with it, asking questions and then following on with others as it presented back results to me were as good, I'd say, as most real-life conversations I've had with data scientists in the past.



The screenshot shows a chat interface with a user and ChatGPT. The user asks for conversion data over six months. ChatGPT provides a three-step process and asks for clarification on event types. The user lists event types. ChatGPT then provides the final result: 83 conversion events.

**User:** Let's expand our definition of converting to anyone who tried to contact us (typically by pressing a Contact Us button), how many conversions did we have over the past six months?

**ChatGPT:** To find the number of conversions over the past six months, we can follow these steps:

1. Define what constitutes a conversion. You mentioned that a conversion is anyone who tried to contact you, which could be identified by a specific event type (e.g., "Contact Attempted" or "Contact Us Button Clicked"). However, the actual event type name depends on your data and how events are tracked. If you could clarify what event type or types represent a contact attempt, I can provide a more accurate count.
2. Filter the data for the past six months.
3. Count the number of conversion events in this filtered data.

However, to proceed, I need to know the exact event type or types that represent a contact attempt in your data. Could you please provide this information?


**User:** any of these event types : Pressed Button, Contact Form Submitted, CTA Pressed, Form Submitted, Cta Pressed

**ChatGPT:** Finished working Show work


Over the past six months, there have been 83 conversion events, according to the expanded definition of conversion (which includes any of the following events: "Pressed Button", "Contact Form Submitted", "CTA Pressed", "Form Submitted", "Cta Pressed").

I then upload a second file containing IP address block-to-geography lookups and check that the Code Interpreter plugin knows what the data in the file is for.

 what do you think this file contains?

 geoiip2-ipv4.csv

---

 Let's open the file and inspect its contents to determine what it contains. 📄 👍 💬

Finished working Show work ▾


The file appears to be a CSV file containing data related to IP geolocation, specifically for Pv4 addresses. Here's a summary of the first few rows:


network	geoname_id	continent_code	continent_name	country_iso_code	country
41.74.160.0/20	49518.0	AF	Africa	RW	Rwanda
41.77.160.0/22	49518.0	AF	Africa	RW	Rwanda
41.138.80.0/21	49518.0	AF	Africa	RW	Rwanda
41.186.0.0/16	49518.0	AF	Africa	RW	Rwanda
41.197.0.0/16	49518.0	AF	Africa	RW	Rwanda

The dataset contains the following columns:

- network:** This seems to represent the IP address range in CIDR format.
- geoname\_id:** This could be an identifier related to a geographic naming standard, such as Geonames.

Then I ask it to combine the geography lookup data with the events data it already has and give me a breakdown of conversions by country.

 using the contents of the two files, give me a geographic breakdown and data visualization of users who have converted

 To provide a geographic breakdown of users who have converted, we'll need to do the following: 📄 👍 💬

First, we will extract the IP addresses from the user events file.

Second, we will match these IP addresses with the geolocation data in the second file to determine the geographic location of each user.

Third, we will count the number of conversions per geographic location.

Finally, we will create a data visualization to display the geographic distribution of conversions.

Finished working Show work ▾

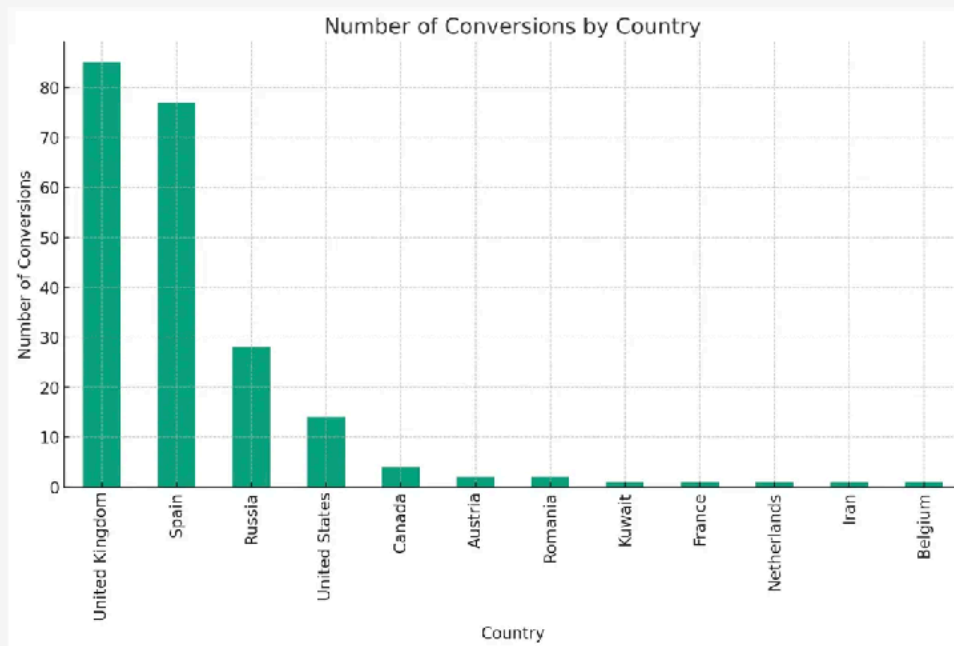
Here is a geographic breakdown of users who have converted:

- United Kingdom: 85 conversions
- Spain: 77 conversions
- Russia: 28 conversions
- United States: 14 conversions
- Canada: 4 conversions
- Austria: 2 conversions
- Romania: 2 conversions
- Kuwait: 1 conversion
- France: 1 conversion
- Netherlands: 1 conversion
- Iran: 1 conversion
- Belgium: 1 conversion

Let's create a bar chart to visualize this data.

Finished working

Show work ▼



Here's a bar chart visualizing the number of conversions by country. The x-axis represents different countries, and the y-axis represents the number of conversions.

As we can see, the United Kingdom and Spain have the highest number of conversions. Other countries like Russia, the United States, and Canada also have significant numbers of conversions, but less than the UK and Spain. A few countries have only a single conversion.

Where the Code Interpreter Plugin did fall-apart, at least in my experience, was when I started asking it to do proper data science-type tasks such as creating a model to predict whether a user was likely to convert.

In my instance it kept having to circle-back because columns it needed weren't in its existing dataframe, then the number of feature values were too many for model type it wanted to use and then finally it ran out of memory and kept asking me to re-upload the two files, forgetting where it had put them and asking me once more to re-upload them.

looking at all the data you have available, what are the best predictors of whether a user coming to our website is likely to convert?

To determine which features are the best predictors of conversion, we can perform a feature importance analysis using a machine learning model such as logistic regression or a tree-based model like random forest.

However, before we can do this:

- Feature Encoding:** Many of the features like `'page_title'`, `'country_name'` are categorical. We need to convert these categories into numerical values using various encoding techniques like one-hot encoding.
- Handling Missing Values:** We need to check for missing values in our data. Depending on the nature of the missing values, we can use different methods like regression imputation or mean imputation.
- Feature Selection:** Not all features are equally important. We should select the most relevant features for our model.

conversion. from the mo the most im

Once the dat examine the

Please note t and problem, should be val are robust an

Do you want preprocessing

yes

Great! Let's start by handling missing values and encoding categorical features. For simplicity, we'll fill missing values with a suitable placeholder value depending on the type of the column. For categorical features, we'll use one-hot encoding, which creates new binary columns for each category of each feature.

We'll then use a Random Forest model to analyze the mix of numerical and categorical features.

Let's start with data preprocessing. We'll iterate through each user's session, so we'll need to group the data based on whether any of the features are missing. For simplicity, we'll avoid creating too many new features.

Note that this is a simplified version of the code.

I apologize for the confusion. It appears that `'blended_user_id'` is also not present in the original dataframe. In that case, it won't be possible to compute the number of sessions per user.

Given these constraints, let's consider only the `'event_type'` and `'page_title'` features. We won't be able to include country information, as it seems to be missing from the data we have.

Let's update our preprocessing to include these features and train the model again.

It seems that we've run out of memory while trying to train the model. This can occur when the dataset is too large to fit into the available memory, which can be exacerbated when using methods like one-hot encoding that can greatly increase the number of features.

There are a few ways we could try to address this:

- Reduce dimensionality:** We could try reducing the number of features by selecting the most important ones, or by excluding them from the analysis.
- Feature selection:** We could perform feature selection before training the model. This could be done using methods like recursive feature elimination or LASSO regression.
- Use a simpler model or change the model:** We could try using a simpler model like logistic regression, or we could change the model to a more memory-efficient one like a gradient boosting machine.

Let's try running the code block again. If the problem persists, we may need to explore alternative approaches.

I apologize for the confusion. It seems that the state reset cleared our previous data and variable definitions. We'll need to reimport the necessary libraries, reload the data, and redo the preprocessing steps before we can train the model.

However, considering the persistent memory issues we're experiencing, I think it would be better to use a more memory-efficient model, such as logistic regression. Logistic regression is a simpler model that is generally more memory-efficient than a random forest. Moreover, logistic regression is a good choice for binary classification problems like this one.

Before we proceed with this approach, I'd like to make sure it aligns with your needs. Would you like to continue with logistic regression?

Even so, it was pretty impressive and the problems I hit are no doubt easy to fix either by my reading the documentation or OpenAI upping some memory limits for the Code Interpreter plugin.

As of now all of these generative AI tools for data analysis, data engineering and analytics engineering all suffer from the daleks-can't-climb-stairs problem that a normal human analyst would work around instinctively but crucially, a well labelled and structured data model designed for use by business users will also be easily understood by a text-based LLM.

So can ChatGPT and its various plugins and API ecosystem partners play the role of a data analyst or even replace one?

Realistically its too early to say, but the conversational approach these plugins use is probably better suited to data analysis and business questions than data engineering and we already have clients that are looking to use products such as Delphi rather than hire new data analysts in the future.

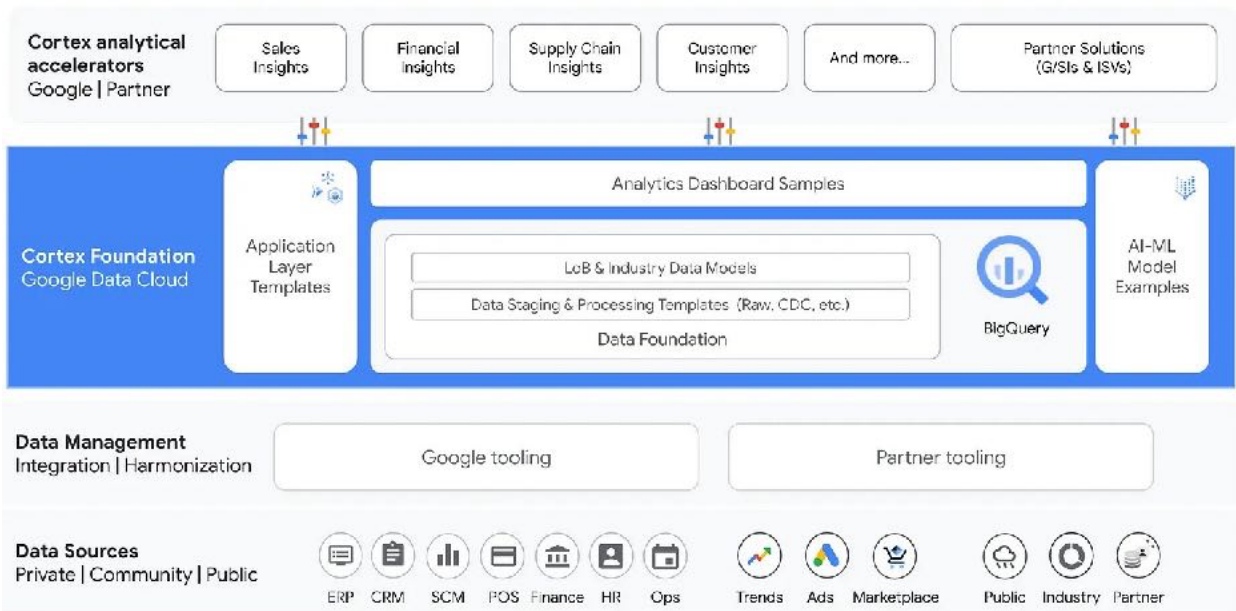


# Google Cloud Cortex Framework brings Packaged Analytics to the Modern Data Stack

Mark Rittman, September 2023

**G**oogle Cloud Cortex Framework is a new packaged analytics initiative from Google Cloud Platform that provides pre-build data extractors, data transformations and interactive dashboards for SAP, Salesforce and a number of marketing and advertising data sources.

Built around Google Cloud Platform technologies such as Google BigQuery, Looker, Google Cloud Composer and Vertex AI, the Google Cloud Cortex Data Foundation element of the framework is a set of reference architectures, building blocks and templates for organizations looking to modernize with Google Cloud Platform's Data Cloud.



Even more interestingly, Google Cloud Cortex Framework is free-to-use, extend and fork and has been made available as an open-source Cortex Data



Foundation repository hosted on Github that contains data transformation logic, data models and templates designed to be deployed as-is or extended by partners and the wider community.

The wider set of Cortex Foundation assets include

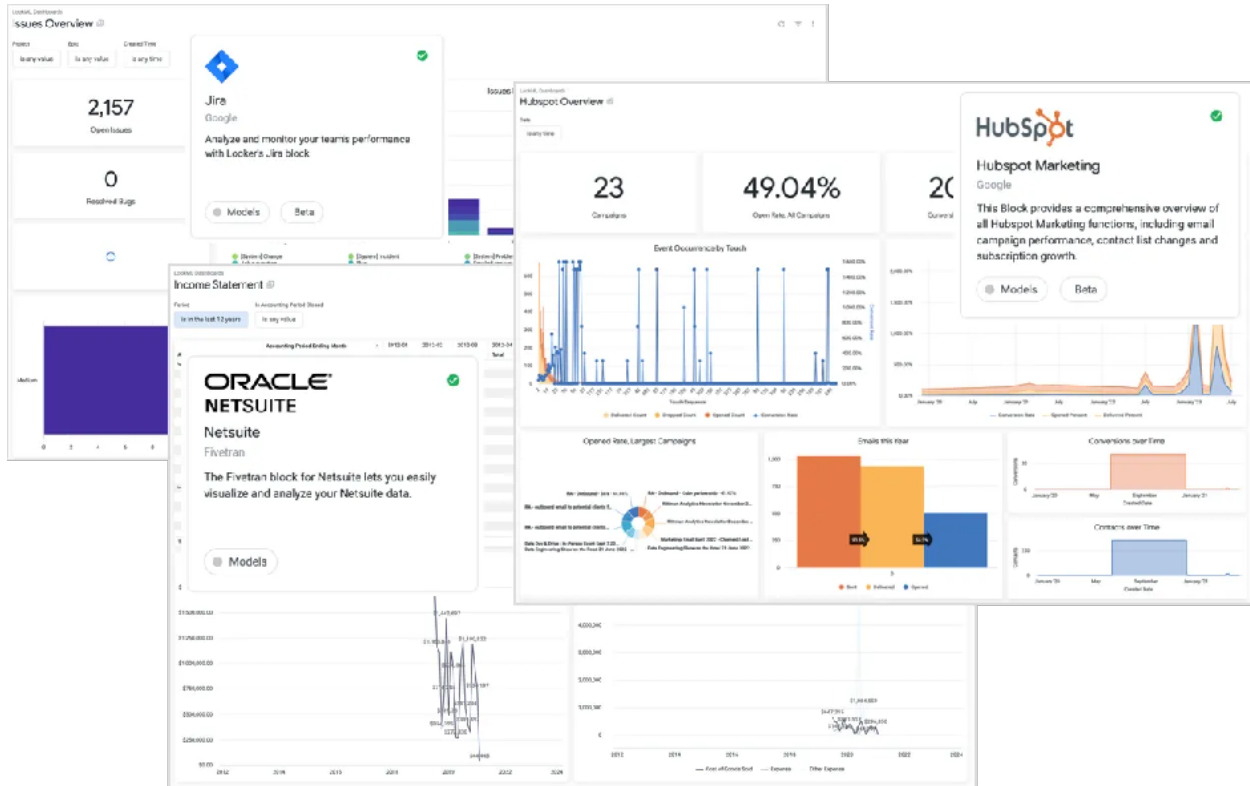
- predefined BigQuery models for private (for example, SAP and Salesforce), public (for example, trends) and commercial (for example, Analytics Hub) data
- data processing templates to transform, enrich and to a limited-extent combine different data sources into cross-data source datasets
- example ML code with Vertex AI
- sample dashboards and explores for use with Looker, such as the one below from the SAP dashboard pack

Benefits to customers making use of these packaged analytics solutions include

- decreased time-to-value compared to building from scratch
- immediate access to best-practice industry KPIs and dashboards
- an end-to-end solution that's pre-integrated and runs at-scale

So What's in this for Google? While Google Cloud Cortex Framework is free to use and adapt for your own use, the payoff from Google's perspective is the SAP, Salesforce and other reporting and hosting workloads it will hopefully make easier to bring over to Google Cloud Platform.

Google Cloud, through the [Looker Marketplace](#) and Looker blocks, has provided a limited form of packaged analytics solutions for Looker customers such as those in the screenshots below for [Jira](#), [NetSuite](#) and [Hubspot](#).



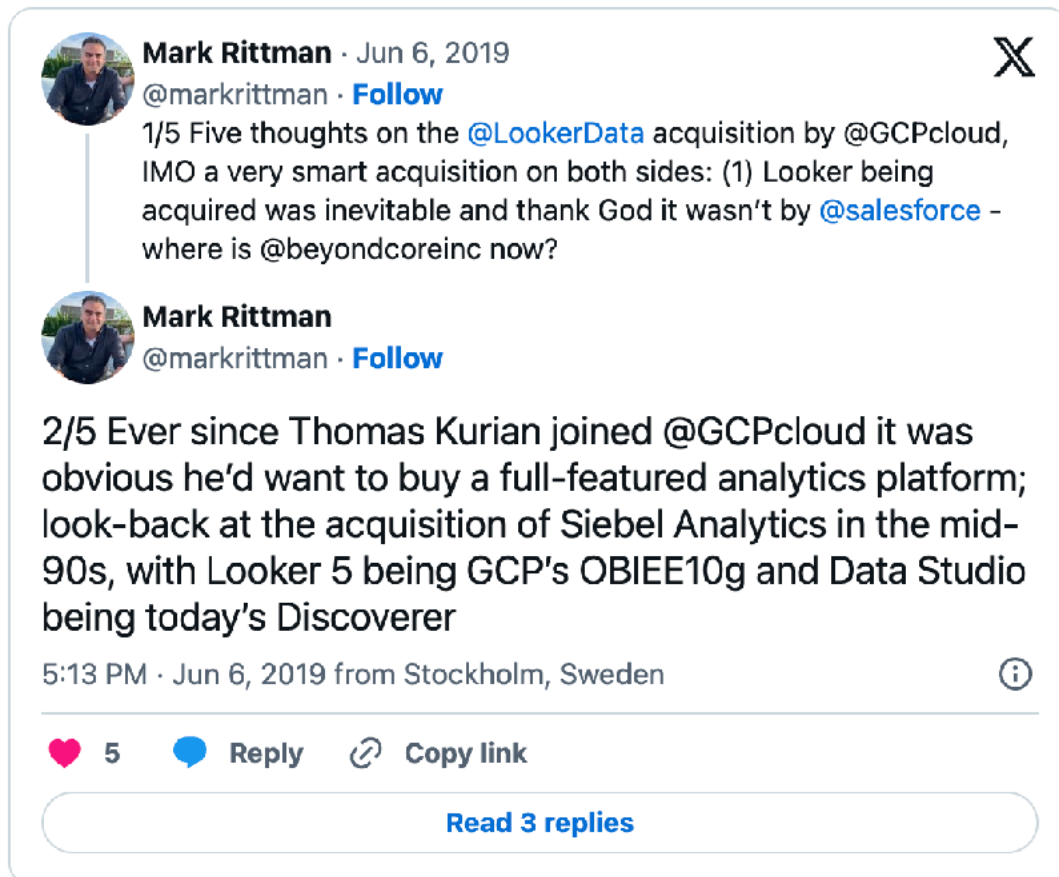
We've also published Looker blocks in the past for uses-cases such as [Multi-touch](#), [Multi-Cycle Marketing Attribution for Google Analytics 4](#) but they aren't easily customisable and still leave you with silos of reporting rather than an integrated, single view of all your operations.

But ever since the news broke back in 2018 that [Thomas Kurian was leaving Oracle to join Google and head-up Google Cloud](#), something like this (and the Looker acquisition that preceded it) was always on the cards. [From my blog written at the time:](#)

“5. Expect to see GCP moving increasingly into packaged SaaS application and analytics solutions for ERP, CRM and Financials to complement their commoditised IaaS and PaaS cloud business and leveraging their massive GSuite and increasingly ChromeOS install base ... and a business model that could provide these applications and packaged analytics for free... That's

the real existential threat to Oracle; spending all their time trying to win an un-winnable cloud infrastructure war and then GCP coming along and making ERP, CRM and business applications and their analytics essentially free.”

and [on Twitter](#):



**Mark Rittman** · Jun 6, 2019 X  
@markrittman · [Follow](#)  
1/5 Five thoughts on the [@LookerData](#) acquisition by [@GCPcloud](#), IMO a very smart acquisition on both sides: (1) Looker being acquired was inevitable and thank God it wasn't by [@salesforce](#) - where is [@beyondcoreinc](#) now?

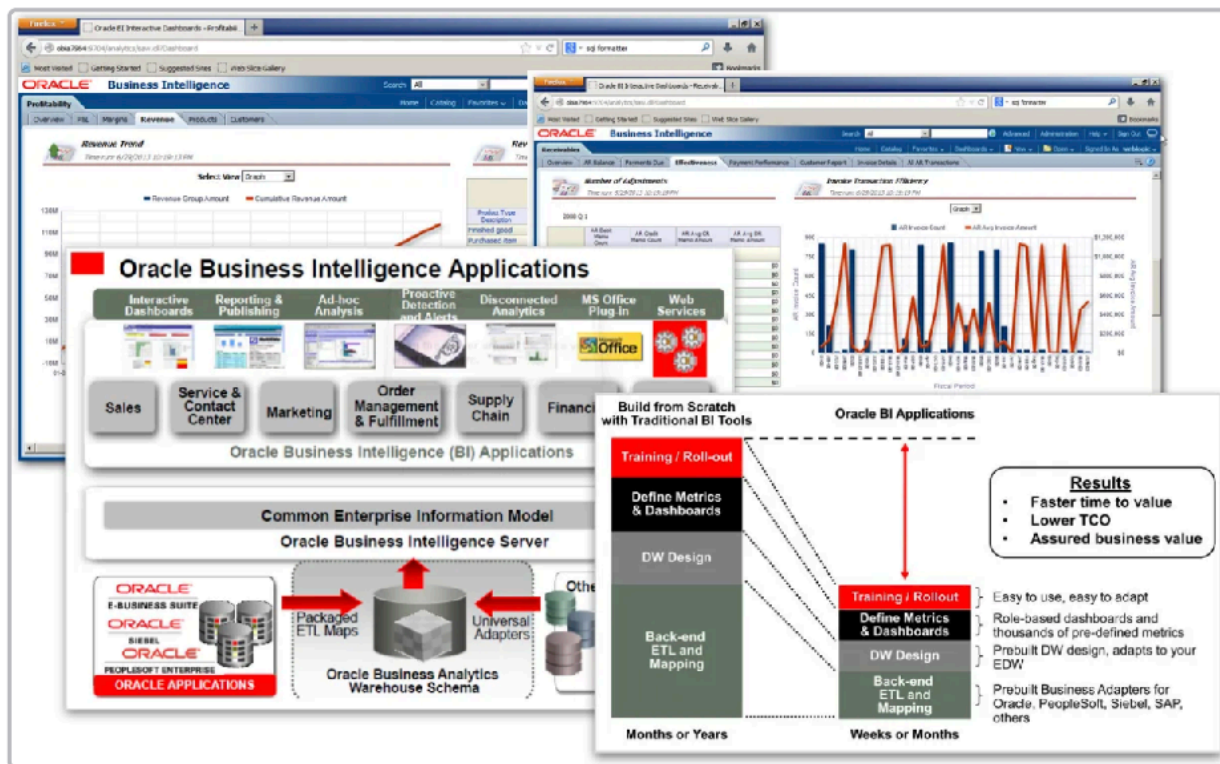
**Mark Rittman**  
@markrittman · [Follow](#)  
2/5 Ever since Thomas Kurian joined [@GCPcloud](#) it was obvious he'd want to buy a full-featured analytics platform; look-back at the acquisition of Siebel Analytics in the mid-90s, with Looker 5 being GCP's OBIEE10g and Data Studio being today's Discoverer

5:13 PM · Jun 6, 2019 from Stockholm, Sweden i

♥ 5    [Reply](#)    [Copy link](#)

[Read 3 replies](#)

Pre-built analytics solutions for Oracle and other vendors' ERP, CRM and Financials applications, running on-top of Oracle's suite of database and analytics tools, were always super-popular with Oracle's customers as – in theory at least – the hard work required to build an integrated data warehouse and set of dashboards had already been done for you, bringing down the time to deploy an enterprise reporting solution from months to weeks.

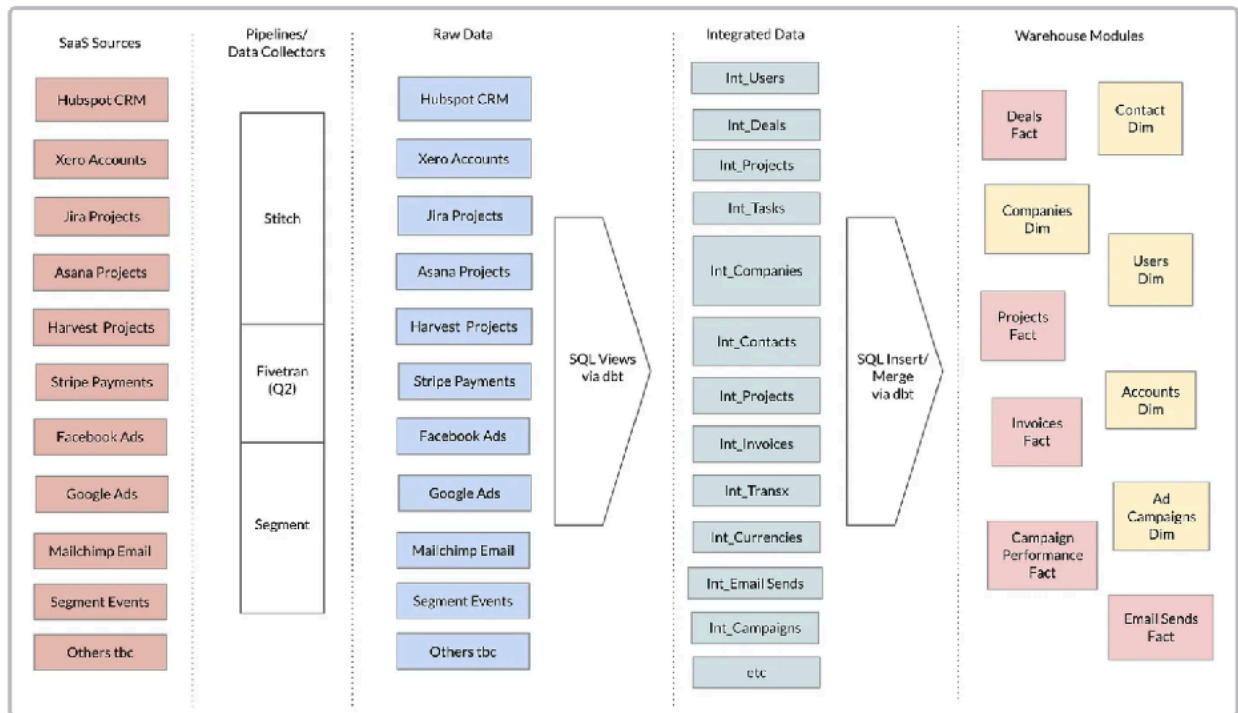


The value delivered to customers using this buy vs. build approach to enterprise data warehousing was such that the license cost of each individual application – Sales Analytics, Financial Analytics, Customer Analytics and so on – were typically priced at 2–3x the price of the analytics tools they ran on.

People buy solutions, not technology, as the saying goes, and packaged analytics solutions built on pre-built and integrated data warehouses were (and still are) a key part of Oracle’s appeal to the enterprise market – the market that Google had hired Thomas Kurian in help them break-into.

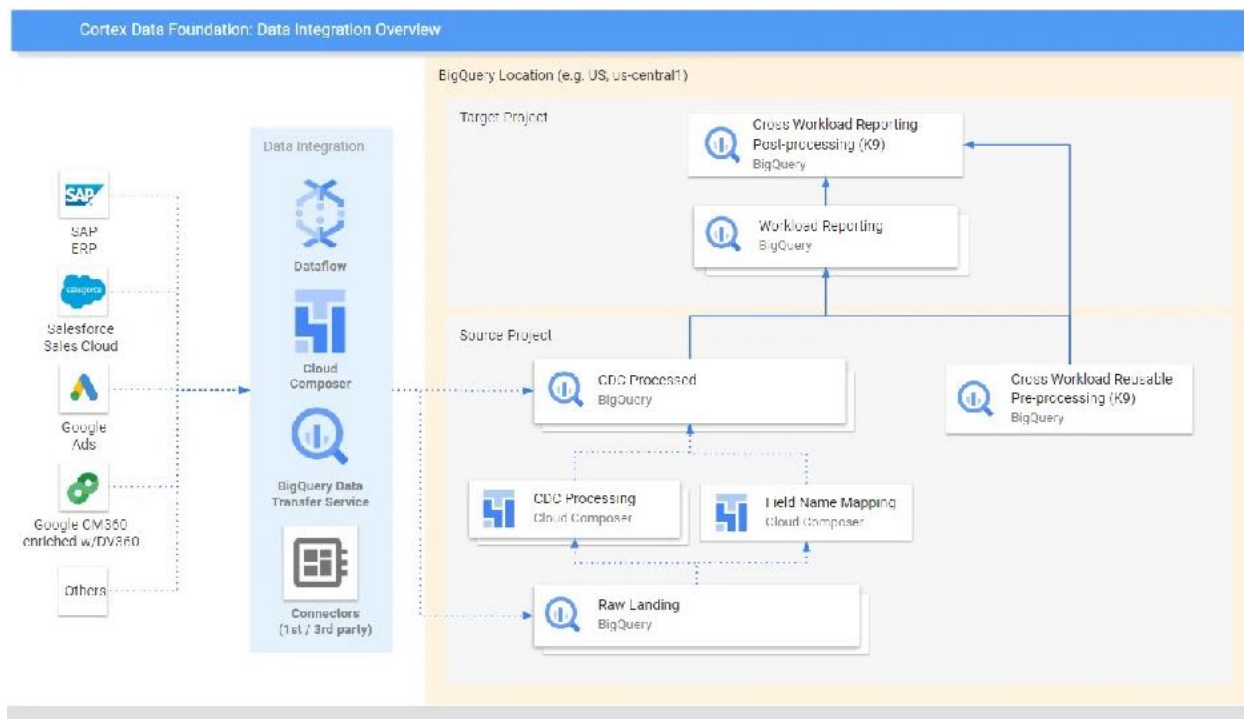
And if you’re a long-term reader of this blog you’ll probably be aware of Rittman Analytics’ take on packaged analytics and data warehousing, our dbt package and open-sourced warehousing toolkit called RA Warehouse for dbt.

It uses a similar design approach to Oracle BI Applications’ packaged data warehouse but focuses instead on SaaS application sources such as Hubspot, Xero and Jira, uses updated modern data stack technology such as Google BigQuery, dbt and Looker and creates a single integrated view of your business across key subject areas.



So how does Google Cloud Cortex Data Foundation compare to these packaged analytics approaches, and how can our RA Data Warehouse and other partner data sources and BI tools integrate with and extend Google’s framework?

Google Cloud Cortex Foundation, as you’d expect from an initiative designed to drive usage and consumption of Google Cloud Platform services, uses Google Cloud Platform tools and services to enable data integration, processing, analytics and reporting.

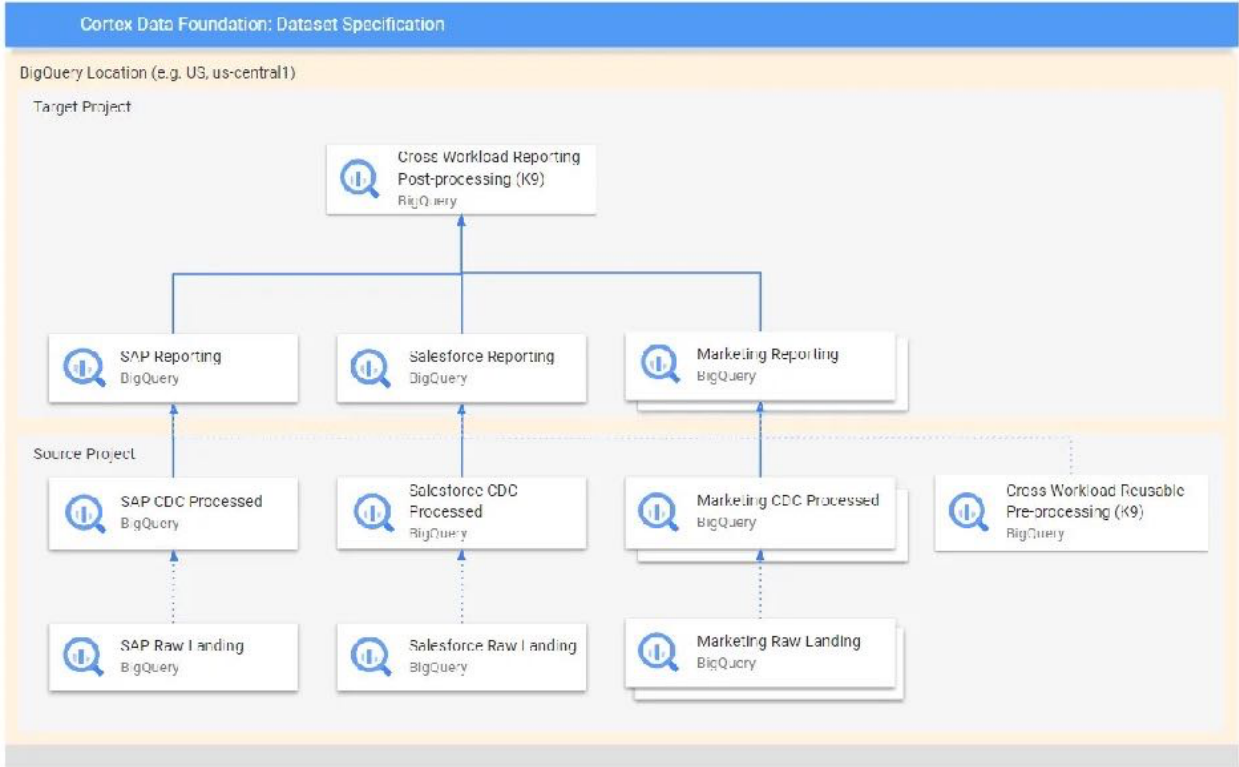


- Google BigQuery is used as the main data warehouse to store raw replicated data, processed change data capture tables and reporting datasets generated from analytical models.
- Google Cloud Storage buckets are used to store files generated during deployment like DAG scripts, SQL queries and temporary processing files.
- Google Cloud Build is used to trigger deployment processes that execute steps to deploy datasets, views, models and other artifacts to BigQuery.
- Google Cloud Composer, rather than dbt or more surprisingly, Dataform, is the data integration and orchestration tool used to transform data and schedule change data capture processing through custom DAGs
- Google Cloud Dataflow is used as the data extraction and data pipeline tool to replicate data from Google Ads into BigQuery, for example
- Google Secret Manager stores credentials needed to connect deployment scripts at runtime to data sources such as Salesforce
- Identity and Access Management provisions and controls access to resources through roles, permissions and service accounts

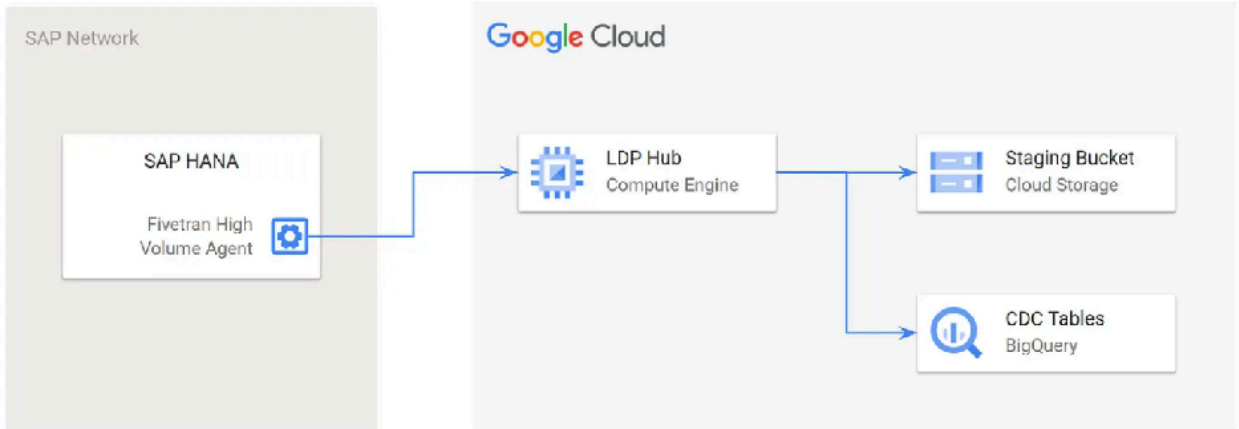
At this point in time, the supported data sources for Cortex Data Foundation are SAP, Salesforce and a number of marketing sources. Each data source has its own method of data extraction, for example:

- SAP ECC or SAP S/4HANA transactional data can be streamed into BigQuery using partner tools such as SnapLogic, Boomi, Informatica Cloud etc; if real-time replication of SAP data is not required, scheduled exports from SAP can be loaded into BigQuery tables on a periodic basis with change data capture (CDC) processing then applied to surface only the latest version in separate CDC datasets
- For Salesforce data, the deployment process sets up a connected app in Salesforce for authentication, and then raw data is then extracted from Salesforce APIs in real-time with CDC processing updating the latest records.
- Google Ads data is integrated using the Google Cloud Dataflow runner to ingest raw campaign performance data from Google Ads APIs.
- Google Campaign Management 360 (CM360) and other platforms such as TikTok also have replication modules that extract raw APIs/exports into BigQuery for further processing.

Some degree of data centralisation and integration across these sources is provided by the framework into what's termed the "K9" (as in DAGs...) datasets but not to the point where single deduplicated customer records, for example, are created.



But Cortex Data Foundation is part of an open framework and Fivetran, for example, can be used to replicate on-premises database sources such as SAP HANA into BigQuery using Fivetran High Volume Agent, as shown in the diagram before.





Or you can use Fivetran to do what we're currently working on as an extension to Cloud Cortex Framework; integrating our RA Warehouse data centralisation framework and Fivetran's connectors into Cortex Data Foundation to bring in the long tail of other data sources such as Hubspot, Xero, Netsuite and Oracle Fusion HCM and CRM ... and create a single view of customers, companies, products and other key business entities.

The screenshot shows the Google Cloud BigQuery interface. On the left is the Explorer pane showing a project named 'gcp-cortex-demo-aug23' with various datasets. The main pane shows a query editor with the following SQL query:

```
SELECT * FROM `gcp-cortex-demo-aug23.CORTEX_K9_REPORTING.contacts` where contact_name = 'James Smith'
```

The query results are displayed in a table with the following columns: contact\_pk, contact\_name, contact\_nature\_keys, and contact\_city. The results show one row for 'James Smith' with multiple contact\_nature\_keys values.

Row	contact_pk	contact_name	contact_nature_keys	contact_city
1	b9e731e6-d211-4a46-9045-015...	James Smith	netsuite-1518 hubspot-2657501 sfdc-0038e00c330UbveAAC	Tucson

# Bringing dbt and Analytics Engineering to Oracle Autonomous Data Warehouse

Mark Rittman, October 2023

**d**bt ([Data Build Tool](#)) is an open source command-line tool for managing and automating data transformation workflows. It enables a standardised, modular approach to building analytics code and is typically used as part of an increasingly popular method of developing data warehouses and data platforms called [analytics engineering](#).

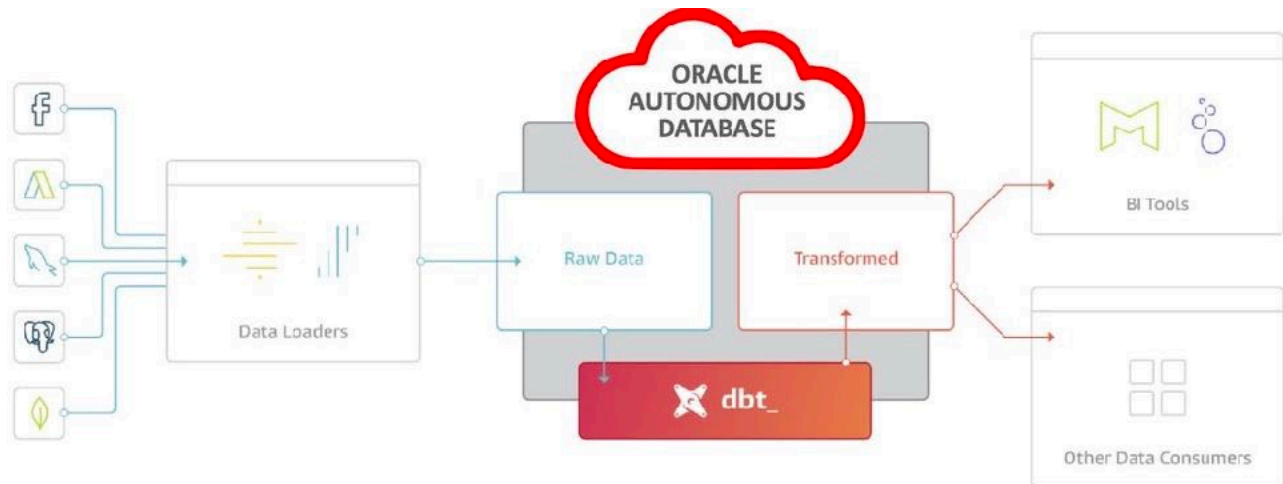
The analytics engineering approach combines software engineering principles with analytics to create more maintainable, trustworthy data pipelines built on transformation logic that is modular, scalable, tested, and well-documented.

With dbt, analytics code progresses from messy, ad hoc scripts to structured engineering workflows developed using a [software development lifecycle](#). In this post, I'll introduce dbt, explain its benefits for Oracle Database and Autonomous Data Warehouse developers and walk through getting started with dbt and the new [dbt-oracle adapter](#).

What is dbt?

ELT (Extract, Load, Transform) is a modern data warehousing pattern very familiar to Oracle data warehouse developers where raw data is first extracted and loaded into the target warehouse, then transformed within the warehouse itself.

This is where dbt comes in – it provides the “T” for managing transformations in ELT, with the E (extract) and L (load) typically handled by services such as [Fivetran](#), Airbyte or Stitch.



Originally developed to help data analysts within the VC-backed startup world develop production-ready data transformation code for Amazon

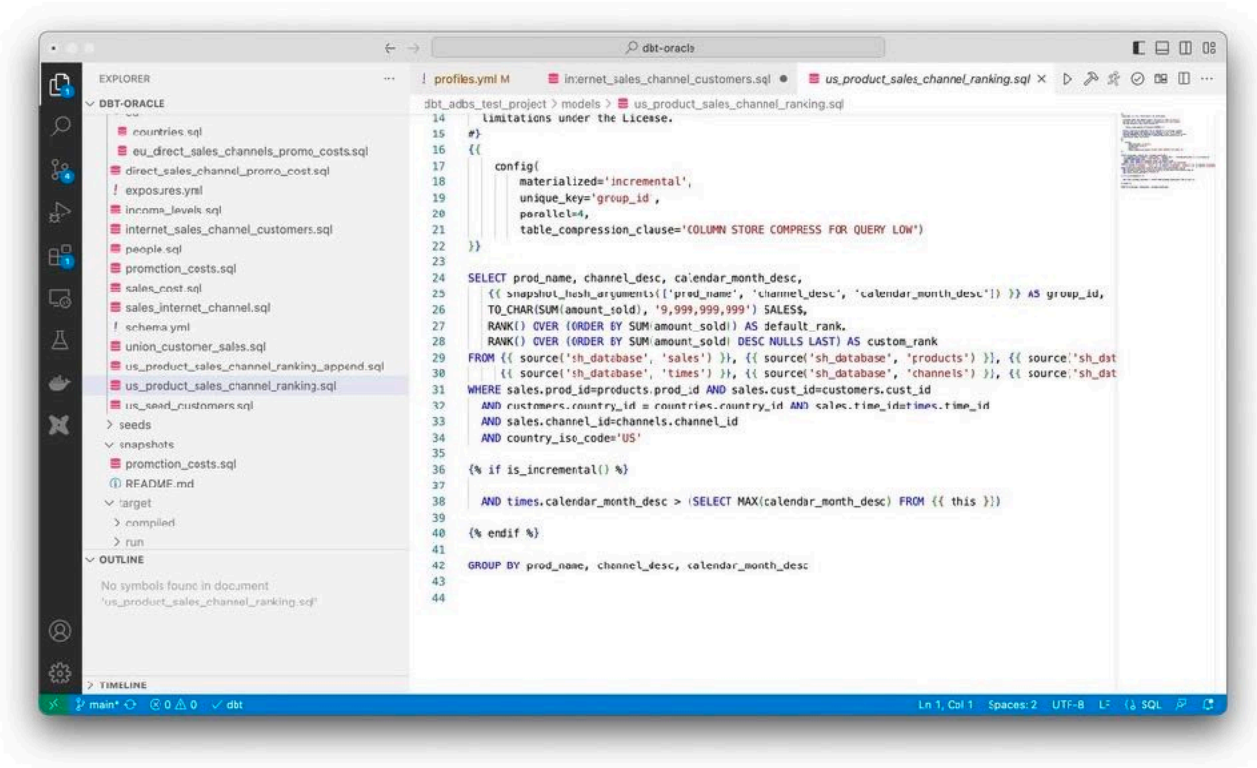
Redshift and other cloud data warehouses, dbt is now more of an ecosystem than single tool and comprises of:

- An open source core ([dbt Core](#)), originally developed by Fishtown Analytics (now dbt Labs) that you run from the command-line and can schedule using cron jobs or orchestrators such as [Dagster](#)
- a commercial SaaS service ([dbt Cloud](#)) that provides features for developing, testing, scheduling and investigating data models
- community-provided utilities and content packages (such as our [RA Attribution](#) package for multi-touch marketing attribution) along with [official](#) and [related](#) Slack groups

Key capabilities of dbt include:

- Dependency management – dbt handles model dependencies automatically
- Testing – add tests to models to ensure data quality
- Refactoring – easily refactor models without breaking dependencies
- Documentation – auto-generate docs for all models in a project
- CI/CD Integration – build, test, and deploy projects on every code change

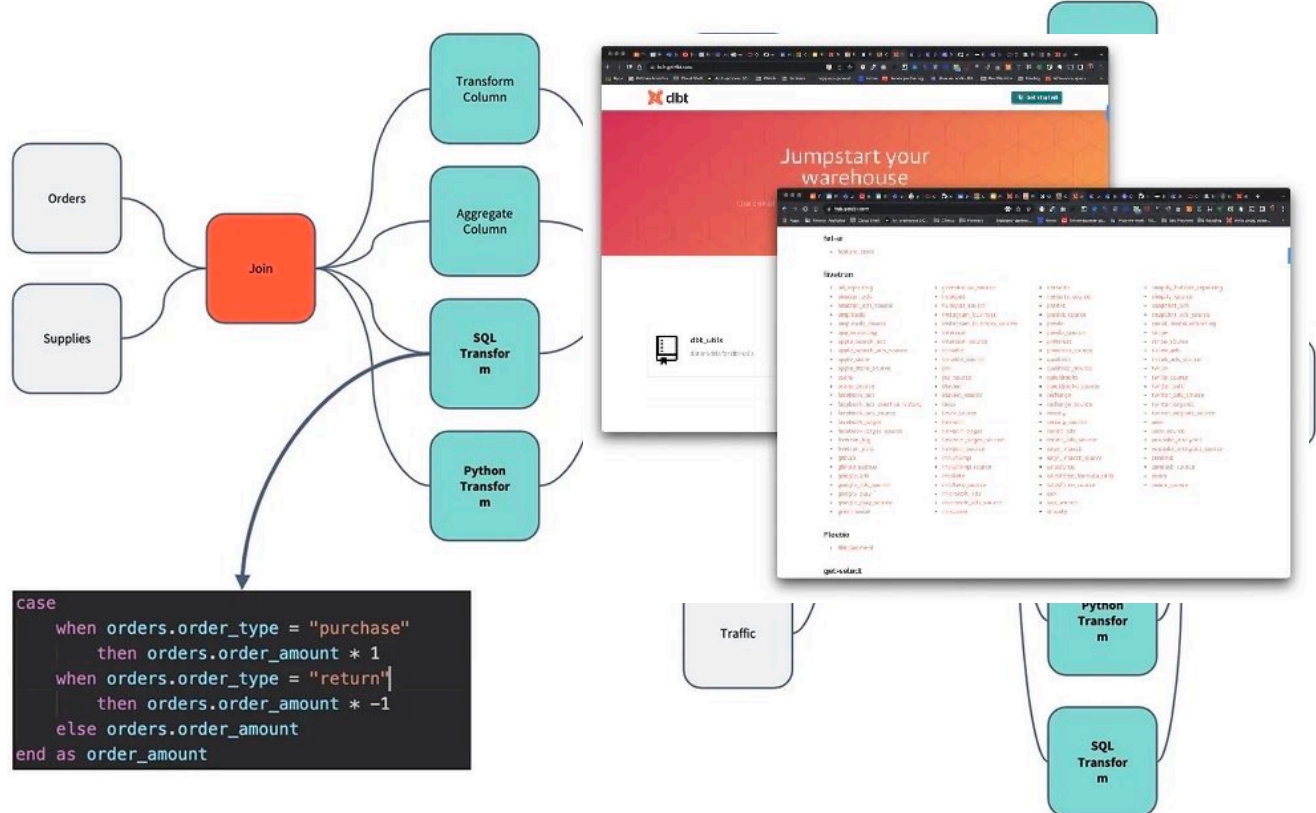
Whereas tools such as Oracle Data Integrator (ODI) and Oracle Warehouse Builder provide a visual, GUI-based mapping interface for designing data flows, dbt instead takes a code-centric approach with transformations encapsulated in modular SQL SELECT statements called “models”, either written and run using a text editor or tool such as Microsoft VS Code.



Developing transformation logic using a GUI has the benefit of opening-up this task to a wider audience, but when the business requires complex data transformations such an approach can become unwieldy.

More of a concern though is the ungoverned shadow SQL or Python code that drag-and-drop tool users end-up adding to their visual transformation nodes, making it hard to track changes and ensure those transformations and embedded are tested and reliable in a scalable and automated way.

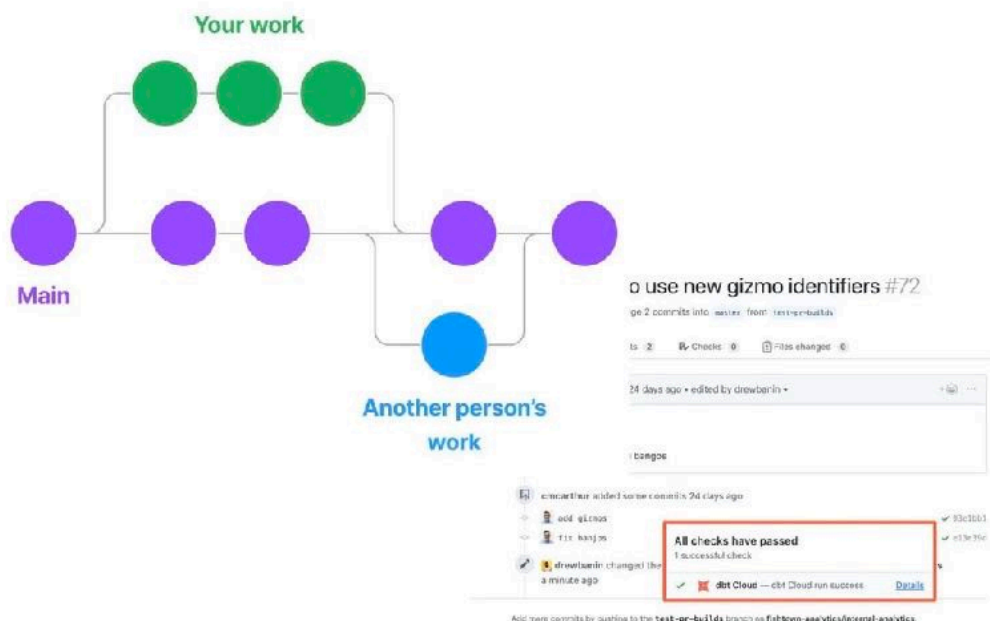
A key difference between ODI and dbt is therefore in its approach to modularity and reusable transformation logic. With ODI, code reuse is achieved by copying and modifying mappings across projects; dbt, however achieves this through the



use of those modular, configurable SQL models that can be packaged up into shared libraries and invoked repeatedly.

Also, crucially, dbt models exist as simple SQL files making them straightforward to version control via Git. ODI's visual mappings are stored within its repository, requiring use of ODI lifecycle tools.

With dbt, developers can utilize the full power of Git for branching strategies, pull requests, and decentralized development.



dbt models also provide visibility into history and differences between versions.

## Getting Started with dbt-oracle

The [dbt-oracle adapter](#) (originally created by Indicium, now supported and maintained directly by Oracle) is a python package that contains the core dbt libraries together with an adapter for connecting to on-premises and cloud Oracle databases. This, along with an install of Python 3 and the [oracledb client drivers package](#) are what's required to install dbt on your Mac, for example.

```
python3 -m venv dbt-oracle-venv source dbt-oracle-venv/bin/activate
pip install --upgrade pip python -m pip install oracledb --upgrade
python -m pip install dbt-oracle
```

The dbt-oracle Github repo at <https://github.com/oracle/dbt-oracle> contains a demo dbt package that transforms data from the SH (Sales History) example schema shipped with all Oracle database versions.

Looking at an example dbt model within that demo package, you can see examples of model configuration settings, references to other tables in the form of sources and some Jinja templating code that adds a further WHERE clause to the SQL statement when the model is compiled in incremental load mode.

```
{{config(materialized='incremental', unique_key='group_id')}}
WITH direct_sales_promo_cost AS (
    SELECT s.prod_id,
           s.quantity_sold,
           s.amount_sold,
           s.time_id,
           c.channel_desc,
           p.promo_name,
           p.promo_cost,
           {{ hash_arguments(['s.prod_id', 's.quantity_sold',
's.time_id', 'p.promo_name']) }} AS group_id
    FROM {{ source('sh_database', 'sales') }} s,
         {{ source('sh_database', 'promotions') }} p,
         {{ source('sh_database', 'channels') }} c
    WHERE s.channel_id = 3
          AND s.promo_id = p.promo_id
          AND s.channel_id = c.channel_id
```

```

    {% if is_incremental() %}
        AND s.time_id > (SELECT MAX(time_id) FROM {{ this }})
    {% endif %}
)
SELECT * FROM direct_sales_promo_cost

```

Other model definitions use [ref\(\) Jinja functions](#) instead of table names to define dependencies between tables in a transformation DAG, and use Oracle-specific configuration settings to tell dbt how specifically to materialize the database object that is the result of compiling this particular model.

```

{{config(
    materialized='table',
    parallel=4,
    table_compression_clause='COLUMN STORE COMPRESS FOR QUERY'
)}}
select
    c.cust_id, c.cust_first_name, c.cust_last_name, t.country_iso_code,
t.country_name, t.country_region
from
    {{ ref('sales_internet_channel') }} s,
    {{ source('sh_database', 'countries') }} t,
    {{ source('sh_database', 'customers') }} c
WHERE s.cust_id = c.cust_id
AND c.country_id = t.country_id

```

dbt Core data source connections are defined using a configuration file in YAML format stored by default at \$HOME/.dbt/profiles.yml, outside of the git repository files so database credentials don't get published with your code by accident.

```

dbt_test:
  target: dev
  outputs:
    dev:
      type: oracle
      user: MARK_DBT
      pass: *****
      protocol: "tcps"
      host: adb.uk-london-1.oraclecloud.com
      port: 1522
      service:
g71ab9f09757846_radevelopmentadw_high.adb.oraclecloud.com

```

```
schema: MARK_DBT
retry_count: 1
retry_delay: 5
shardingkey:
  - skey
supershardingkey:
  - skey
cclass: CONNECTIVITY_CLASS
purity: self
threads: 1
```

There are various ways that dbt can connect to an Oracle database; using thin or thick drivers from the python-oracledb Oracle client libraries package along with the choice of either TLS (Transport Layer Security) or mutual TLS (mTLS) when connecting to Oracle Autonomous Data Warehouse or Autonomous Transaction Processing.

I used the thin client mode and was able to connect to Oracle Autonomous Data Warehouse Cloud without using a wallet file, using just my database username, password and host/port/service name.

dbt Core packages are run from the command-line and when executed, compile your SQL and Jinja model code into SQL SELECT and MERGE statements that transforms, tests and runs utilities to transform your data within the Oracle database.

```
(dbt-oracle-venv) markrittman@marks-imac-4 dbt_adbs_test_project % dbt
build
21:47:47 Running with dbt=1.5.3
21:47:47 oracle adapter: Running in thin mode
21:47:47 Registered adapter: oracle=1.5.3
21:47:47 Found 17 models, 10 tests, 1 snapshot, 1 analysis, 637
macros, 3 operations, 1 seed file, 8 sources, 2 exposures, 0 metrics,
0 groups
21:47:47
21:47:50
21:47:50 Running 1 on-run-start hook
21:47:50 1 of 1 START hook: dbt_adbs_test_project.on-run-
start.0 ..... [RUN]
21:47:50 1 of 1 OK hook: dbt_adbs_test_project.on-run-
start.0 ..... [OK in 0.03s]
21:47:50
```



```

21:47:50 Concurrency: 1 threads (target='dev')
21:47:50
21:47:50 1 of 28 START sql table model
MARK_DBT.countries ..... [RUN]
21:47:52 1 of 28 OK created sql table model
MARK_DBT.countries ..... [OK in 1.30s]
21:47:52 2 of 28 START sql view model
MARK_DBT.direct_sales_channel_promo_cost ..... [RUN]
21:47:53 2 of 28 OK created sql view model
MARK_DBT.direct_sales_channel_promo_cost ..... [OK in 0.88s]
21:47:53 3 of 28 START sql table model
MARK_DBT.promotion_costs ..... [RUN]
21:47:54 3 of 28 OK created sql table model
MARK_DBT.promotion_costs ..... [OK in 1.30s]
21:47:54 4 of 28 START sql table model
MARK_DBT.promotion_costs_for_direct_sales_channel [RUN]
21:47:56 4 of 28 OK created sql table model
MARK_DBT.promotion_costs_for_direct_sales_channel [OK in 2.49s]
21:47:56 5 of 28 START sql incremental model
MARK_DBT.promotion_costs_for_direct_sales_channel_incr_insert [RUN]
21:47:59 5 of 28 OK created sql incremental model
MARK_DBT.promotion_costs_for_direct_sales_channel_incr_insert [OK in
2.37s]
21:47:59 6 of 28 START sql incremental model
MARK_DBT.promotion_costs_for_direct_sales_channel_incr_merge [RUN]
21:48:00 6 of 28 OK created sql incremental model
MARK_DBT.promotion_costs_for_direct_sales_channel_incr_merge [OK in
1.56s]
21:48:00 7 of 28 START sql incremental model
MARK_DBT.promotion_costs_for_direct_sales_channel_incr_merge_unique_ke
ys [RUN]
21:48:02 7 of 28 OK created sql incremental model
MARK_DBT.promotion_costs_for_direct_sales_channel_incr_merge_unique_ke
ys [OK in 1.78s]
21:48:02 8 of 28 START sql table model
MARK_DBT.sales_cost ..... [RUN]
21:48:04 8 of 28 OK created sql table model
MARK_DBT.sales_cost ..... [OK in 1.64s]
21:48:04 9 of 28 START sql table model
MARK_DBT.sales_internet_channel ..... [RUN]
21:48:06 9 of 28 OK created sql table model
MARK_DBT.sales_internet_channel ..... [OK in 2.20s]
21:48:06 10 of 28 START sql table model MARK_DBT.union_customer_sales
..... [RUN]
21:48:12 10 of 28 OK created sql table model

```

```

MARK_DBT.union_customer_sales ..... [OK in 6.14s]
21:48:12 11 of 28 START sql incremental model
MARK_DBT.us_product_sales_channel_ranking . [RUN]
21:48:14 11 of 28 OK created sql incremental model
MARK_DBT.us_product_sales_channel_ranking [OK in 2.20s]
21:48:14 12 of 28 START sql incremental model
MARK_DBT.us_product_sales_channel_ranking_append [RUN]
21:48:17 12 of 28 OK created sql incremental model
MARK_DBT.us_product_sales_channel_ranking_append [OK in 2.62s]
21:48:17 13 of 28 START seed file
MARK_DBT.seed ..... [RUN]
21:48:18 13 of 28 OK loaded seed file
MARK_DBT.seed ..... [INSERT 5 in
0.96s]
21:48:18 14 of 28 START test
dbt_constraints_primary_key_countries_country_id ..... [RUN]
21:48:19 14 of 28 PASS
dbt_constraints_primary_key_countries_country_id .....
[PASS in 1.50s]
21:48:19 15 of 28 START test
not_null_countries_country_id ..... [RUN]
21:48:21 15 of 28 PASS
not_null_countries_country_id .....
[PASS in 1.14s]
21:48:21 16 of 28 START test
unique_countries_country_id ..... [RUN]
21:48:21 16 of 28 PASS
unique_countries_country_id .....
[PASS in 0.62s]
21:48:21 17 of 28 START sql view model
MARK_DBT.us_seed_customers ..... [RUN]
21:48:22 17 of 28 OK created sql view model
MARK_DBT.us_seed_customers ..... [OK in 0.90s]
21:48:22 18 of 28 START snapshot
MARK_DBT.promotion_costs_snapshot ..... [RUN]
21:48:24 18 of 28 OK snapshotted
MARK_DBT.promotion_costs_snapshot ..... [success in
2.28s]
21:48:24 19 of 28 START sql table model
MARK_DBT.internet_sales_channel_customers ..... [RUN]
21:48:29 19 of 28 OK created sql table model
MARK_DBT.internet_sales_channel_customers .. [OK in 4.41s]
21:48:29 20 of 28 START sql table model
MARK_DBT.people ..... [RUN]
21:48:32 20 of 28 OK created sql table model

```

```

MARK_DBT.people ..... [OK in 2.86s]
21:48:32 21 of 28 START sql table model
MARK_DBT.eu_direct_sales_channels_promo_costs ... [RUN]
21:48:34 21 of 28 OK created sql table model
MARK_DBT.eu_direct_sales_channels_promo_costs [OK in 1.92s]
21:48:34 22 of 28 START test
accepted_values_people_gender__Male__Female ..... [RUN]
21:48:35 22 of 28 PASS
accepted_values_people_gender__Male__Female .....
[PASS in 1.13s]
21:48:35 23 of 28 START test
dbt_constraints_primary_key_people_id ..... [RUN]
21:48:36 23 of 28 PASS
dbt_constraints_primary_key_people_id .....
[PASS in 0.85s]
21:48:36 24 of 28 START test
not_null_people_id ..... [RUN]
21:48:36 24 of 28 PASS
not_null_people_id .....
[PASS in 0.74s]
21:48:36 25 of 28 START test
test_count_employees ..... [RUN]
21:48:37 25 of 28 PASS
test_count_employees .....
[PASS in 0.96s]
21:48:37 26 of 28 START test
unique_people_id ..... [RUN]
21:48:38 26 of 28 PASS
unique_people_id .....
[PASS in 1.05s]
21:48:38 27 of 28 START test
dbt_constraints_foreign_key_eu_direct_sales_channels_promo_costs_count
ry_id__country_id__ref_countries_ [RUN]
21:48:39 27 of 28 PASS
dbt_constraints_foreign_key_eu_direct_sales_channels_promo_costs_count
ry_id__country_id__ref_countries_ [PASS in 0.93s]
21:48:39 28 of 28 START test
relationships_eu_direct_sales_channels_promo_costs_country_id__country
_id__ref_countries_ [RUN]
21:48:40 28 of 28 PASS
relationships_eu_direct_sales_channels_promo_costs_country_id__country
_id__ref_countries_ [PASS in 0.75s]
21:48:41
21:48:41 Running 2 on-run-end hooks
21:48:41 1 of 2 START hook: dbt_adbs_test_project.on-run-

```

```

end.0 ..... [RUN]
21:48:41 1 of 2 OK hook: dbt_adbs_test_project.on-run-
end.0 ..... [OK in 0.03s]
21:48:41 Running dbt Constraints
21:48:45 Creating primary key: COUNTRIES_COUNTRY_ID_PK
21:48:49 Creating primary key: PEOPLE_ID_PK
21:48:53 Creating foreign key: FK_2419558567 referencing countries
['country_id']
21:48:54 Finished dbt Constraints
21:48:54 2 of 2 START hook: dbt_constraints.on-run-
end.0 ..... [RUN]
21:48:54 2 of 2 OK hook: dbt_constraints.on-run-
end.0 ..... [OK in 0.00s]
21:48:54
21:48:54
21:48:54 Finished running 9 table models, 2 view models, 5
incremental models, 1 seed, 10 tests, 1 snapshot, 3 hooks in 0 hours 1
minutes and 7.14 seconds (67.14s).
21:48:54
21:48:54 Completed successfully
21:48:54
21:48:54 Done. PASS=28 WARN=0 ERROR=0 SKIP=0 TOTAL=28

```

dbt packages typically include descriptions for fields, schema definitions and other metadata that, together with the DAG dependency graph defined by all of the ref() and source() definitions within your models can be served-up from the command line using a built-in webserver, like this:

```

(dbt-oracle-venv) markrittman@marks-imac-4 dbt_adbs_test_project % dbt
docs generate
21:54:57 Running with dbt=1.5.3
21:54:57 oracle adapter: Running in thin mode
21:54:57 Registered adapter: oracle=1.5.3
21:54:57 Found 17 models, 10 tests, 1 snapshot, 1 analysis, 637
macros, 3 operations, 1 seed file, 8 sources, 2 exposures, 0 metrics,
0 groups
21:54:57
21:55:00 Concurrency: 1 threads (target='dev')
21:55:00
21:55:03 Running dbt Constraints
21:55:03 Finished dbt Constraints
21:55:04 Building catalog
21:56:12 Catalog written to /Users/markrittman/prod/dbt-oracle/
dbt_adbs_test_project/target/catalog.json
(dbt-oracle-venv) markrittman@marks-imac-4 dbt_adbs_test_project % dbt

```

docs serve

22:06:41 Running with dbt=1.5.3

22:06:41 oracle adapter: Running in thin mode

Serving docs at 8080

To access from your browser, navigate to: <http://localhost:8080>

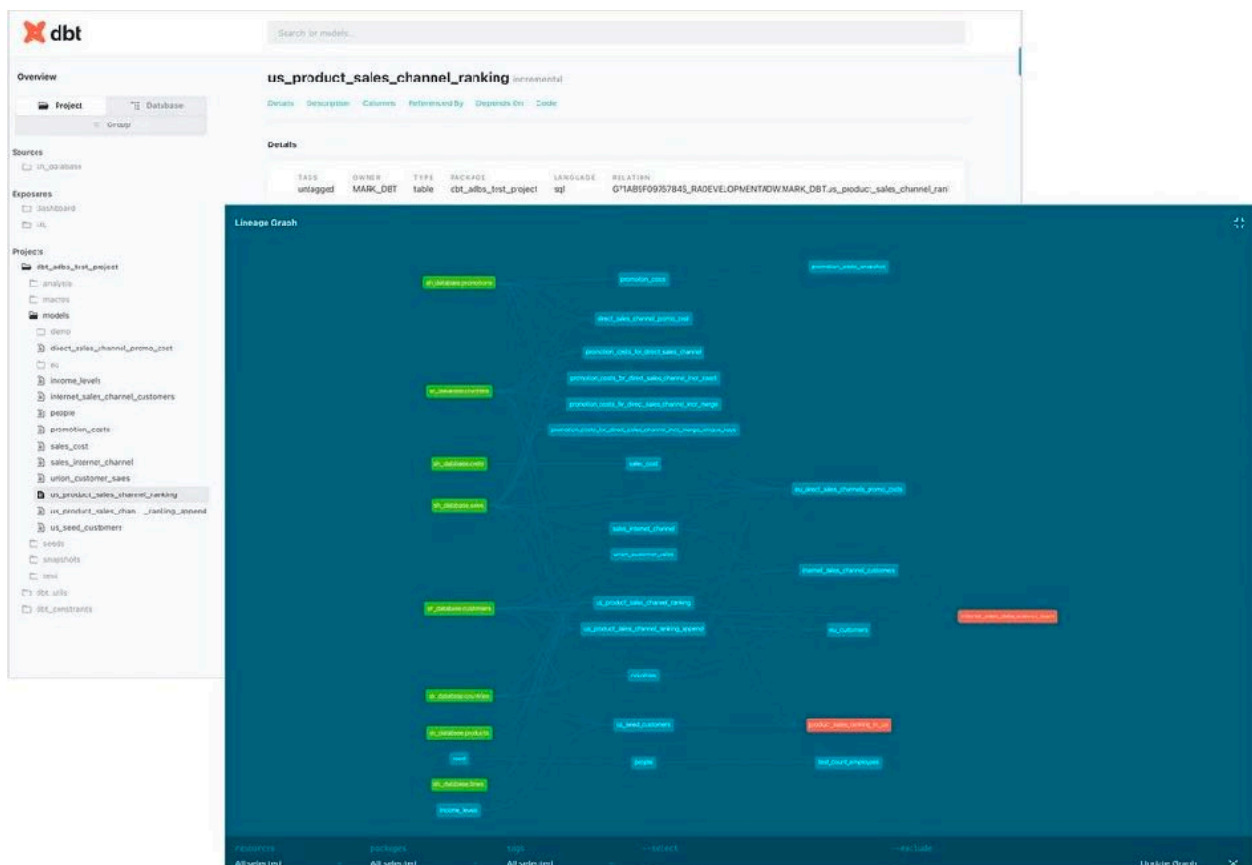
Press Ctrl+C to exit.

127.0.0.1 - - [06/Sep/2023 23:06:41] "GET / HTTP/1.1" 200 -

127.0.0.1 - - [06/Sep/2023 23:06:42] "GET /manifest.json?cb=1694038002092 HTTP/1.1" 200 -

127.0.0.1 - - [06/Sep/2023 23:06:42] "GET /catalog.json?cb=1694038002092 HTTP/1.1" 200 -

and then visualized as a data dictionary portal and transformation dependency graph, like this:



# Unlocking Data Orchestration: How to Orchestrate Fivetran and dbt with Dagster

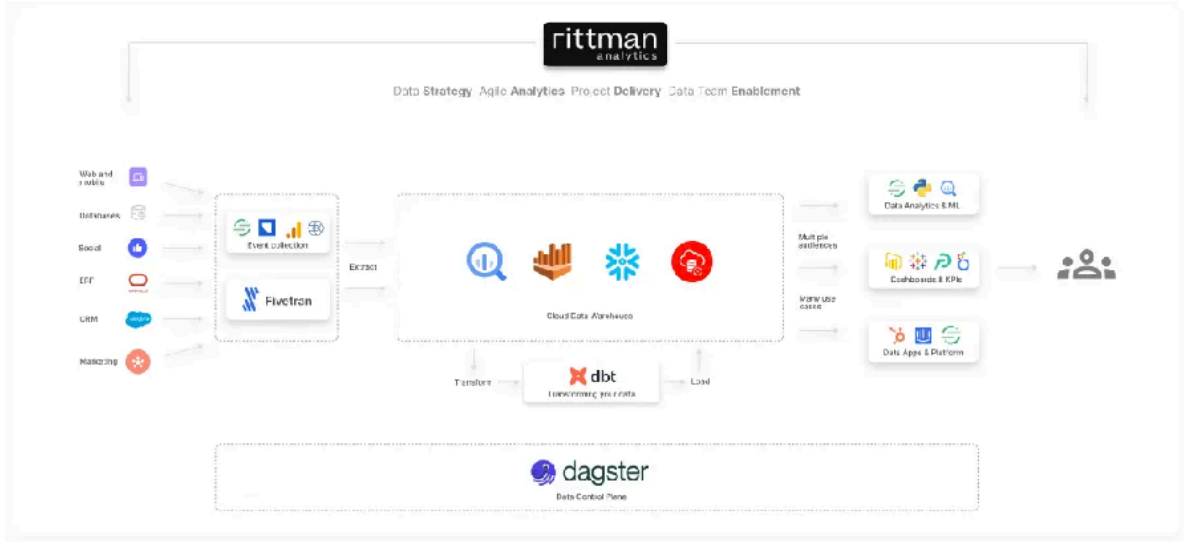
Jordan Ilyat , October 2023

In today's data-centric world, organisations generate volumes of data from various sources, with many tools in the modern data stack creating their own stream of information. Data projects span across multiple teams and systems, leaving organisations without a global view of their most valuable assets.

This fragmented approach to data management results in a complex maze of siloed sources, convoluted transformations, and missed opportunities. The key to informed decision-making lies in the ability to harness the full potential of your data.

Imagine a world where data flows seamlessly from source to analysis, where transformations are executed with precision, and where data engineers and analysts collaborate effortlessly.

This is the world of orchestrating Fivetran and dbt with Dagster – an ecosystem that unlocked to me the power of Dagster, focusing on the end products of data transformation, rather than the process itself.



In a well-orchestrated modern data stack, visibility extends across teams, ensuring that valuable insights are accessible throughout the organisation's data landscape. This accessibility ensures that opportunities for data-guided decision-making are not overlooked, ultimately contributing to the success of strategies and outcomes.

## The Orchestration Journey

Data is an invaluable asset, but its value to an organisation remains untapped until it is harnessed, transformed, and made accessible to those who need it most.

The orchestration journey involves a series of steps, from the moment data is ingested to when it is finally transformed into meaningful insights.

- Fivetran: Streamlining Data Ingestion

“Fivetran solves the data engineering problem of ensuring data flows effortlessly into your centralised data warehouse.”

- dbt: Transforming Data with Confidence

“dbt enables you to transform your raw data into structured, organised, and reliable data.”

- Dagster: Orchestrating the flow

“Dagster is experiencing an evolution, transitioning from an orchestration tool to becoming a true data control plane — a pivotal shift in the role of data orchestrators within the Modern Data Stack”

These three components — Fivetran, dbt, and Dagster — work in harmony to streamline data operations, enable confident data transformations, and orchestrate the flow of software-defined assets throughout the organisation.

## How do you orchestrate Fivetran and dbt with Dagster?

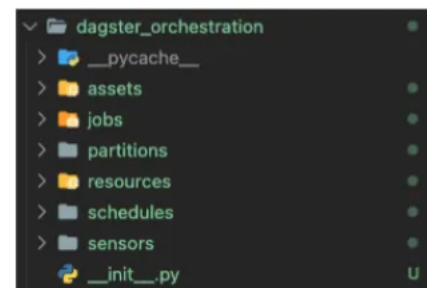
Pre-requisites: you're already using Fivetran, dbt and Dagster Cloud Serverless deployment.

### Step 1: Structuring your Dagster project

When setting up your Dagster project, it's essential to consider ease of navigation, especially for new developers joining your team. A well-organised project structure enhances the developer experience and accelerates onboarding.

We've found that structuring your project with the following format is highly effective:

- Each folder contains a module (indicated by an `__init__.py` file).
- These submodules are then imported into the top-level Dagster orchestration module.



Dagster project structure

This approach not only simplifies the project's architecture but also ensures that developers can quickly locate and access the components they need for their tasks. It fosters a smooth onboarding experience and promotes efficient collaboration within your team.



## Step 2: Managing your Fivetran assets

When integrating Fivetran into your Dagster project, you have two options: you can either load all your Fivetran assets or explicitly declare the specific assets you require. We prefer the latter approach, as it enhances governance over the assets your package syncs.

To implement this approach effectively, follow these steps:

1. Add the “dagster-fivetran” package to your setup.py file
2. In your dagster project, create a dedicated file for your Fivetran assets, such as ‘dagster\_orchestration/assets/fivetran\_sources.py’

```
from dagster import EnvVar, with_resources
from dagster_fivetran import FivetranResource, build_fivetran_assets

# TODO: create fivetran API key and secret and add to Dagster environment variables
fivetran_instance = FivetranResource(
    api_key=EnvVar("FIVETRAN_API_KEY"),
    api_secret=EnvVar("FIVETRAN_API_SECRET"),
)

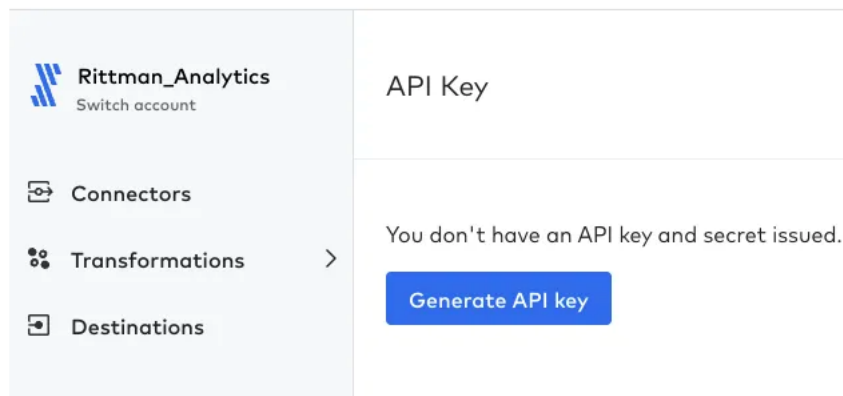
# Fivetran assets
fivetran_assets_google_ads = with_resources(
    build_fivetran_assets(
        connector_id="your_fivetran_connector_id",
        destination_tables=[
            "fivetran_google_ads.account_history",
            "fivetran_google_ads.account_stats",
            "fivetran_google_ads.ad_group_criterion_history",
            "fivetran_google_ads.ad_group_history",
            "fivetran_google_ads.ad_group_stats",
            "fivetran_google_ads.ad_history",
            "fivetran_google_ads.ad_stats",
            "fivetran_google_ads.campaign_history",
            "fivetran_google_ads.campaign_stats",
            "fivetran_google_ads.keyword_stats",
        ],
    ),
    # Use the fivetran_instance resource defined above
    {"fivetran": fivetran_instance},
)
```

This strategy not only provides more control over your defined assets but also simplifies the management of your Fivetran integration within your Dagster project.

### Step 3: Generate Fivetran API credentials

Dagster relies on the Fivetran API to synchronise connectors seamlessly. To establish this connection, you'll need to generate API credentials and incorporate them into your Dagster environment variables. Here's how to do it:

1. **Generate API Credentials:** Visit the Fivetran API configuration page by navigating to <https://fivetran.com/dashboard/user/api-config>. There, you can generate the necessary API key and secret.
2. **Incorporate Credentials:** Once you have your API key and secret, add them to your Dagster environment variables. This integration ensures that Dagster can securely communicate with Fivetran to synchronize your connectors and data seamlessly.



## Step 4: Dagster jobs

In Dagster, jobs serve as the primary unit of execution and monitoring. The essential components of a job are its name and the asset selection to materialise.

Here are the steps to create a job within your Dagster project:

1. Organise your job module: Begin by organising your job module within your Dagster project. Create a dedicated file for your job module, such as 'dagster\_orchestration/jobs/\_\_init\_\_.py'
2. Import necessary components: In this module, import the required components and modules containing your Fivetran and dbt assets. This step ensures that your job has access to the necessary resources for execution.

It's worth noting that these jobs provide the flexibility to update dbt or Fivetran independently. This means you can trigger updates directly from the user interface, offering you control over your data pipeline in one place (Dagster).

```
from dagster import AssetSelection, define_asset_job, load_assets_from_modules
from ..assets import dbt_dw, fivetran_sources

dbt_assets = load_assets_from_modules([dbt_dw])
fivetran_assets = *fivetran_sources.fivetran_assets_google_ads, *fivetran_sources.fivetran_assets_facebook

dbt_dw_job = define_asset_job(
    name="dbt_job",
    selection=AssetSelection.assets(*dbt_assets)
)

fivetran_job = define_asset_job(
    name="fivetran_job",
    selection=AssetSelection.assets(*fivetran_assets)
)

everything_job = define_asset_job(
    name="everything_everywhere_job",
    selection=[*dbt_assets, *fivetran_assets]
)
```

## Step 5. Dagster Schedules

Jobs can be launched in a few different ways. A schedule is a Dagster definition that is used to execute a job at a fixed interval.

Here are the steps to create a schedule within your Dagster project:

1. Organise your schedule module: Begin by structuring your schedule module within your Dagster project. Create a dedicated file for your schedules module, such as 'dagster\_orchestration/schedules/\_\_init\_\_.py'
2. Import necessary components: In this module, import the required components and the module containing your jobs. This step ensures that your schedule has access to the necessary resources for execution.

It's important to note that, in our example, we create a schedule specifically for the 'everything' job. However, additional schedules can be added as needed to accommodate different job execution patterns or intervals.

```
from dagster import ScheduleDefinition
from ..jobs import everything_job

everything_schedule = ScheduleDefinition(
    description="daily project update",
    job=everything_job,
    cron_schedule="30 5 * * *",
)
```

## Step 6: Managing Dependencies

In Dagster, it's essential to establish clear dependencies between your dbt assets and Fivetran assets. This ensures that your data transformations align seamlessly with your data sources. While the approach below is effective, we're always exploring more elegant alternatives.

To manage dependencies, follow these steps:

1. Modify Your dbt Configuration: In your dbt project folder, open the relevant dbt .yml file(s) where you define your models and transformations.
2. Specify Dependencies: Within these files, specify the dependencies between your dbt models and the corresponding Fivetran assets. This step helps Dagster understand the relationships between data sources and transformations.

```
Generate model
- name: ad_group_history
  description: Each record represents a version of an ad group in Google Ads.
  identifier: "{{ var('google_ads_ad_group_history_identifier', 'ad_group_history') }}"
  meta:
    dagster:
      asset_key: ["fivetran_google_ads", "ad_group_history"]
  columns:
    - name: id
      description: "{{ doc('ad_group_id') }}"
    - name: updated_at
```

### Step 7: Definitions object

In Dagster, the Definitions object unifies the core concepts of Dagster. To finalise your orchestration project, follow these steps:

1. Organise your project module: Start by organising your project effectively. Create a dedicated file for your project module, such as dagster\_orchestration/\_\_init\_\_.py'
2. Import necessary components: Within this module, import all the essential components and sub-modules containing your assets, resources, jobs, and schedules. This step ensures that your project has access to the required resources and configurations.
3. Aggregate your definitions: You might have numerous components within your project. To improve readability and maintain a structured codebase, consider aggregating your project definitions.
4. Define the Definitions object: At the heart of your orchestration project is the Definition object. It brings together all the key components and configurations, providing a clear and cohesive view of your data pipeline.

```

from dagster import Definitions, load_assets_from_modules
# Submodules
from .assets import dbt_dw, fivetran_sources
from .resources import dbt_project_resource
from .jobs import dbt_dw_job, fivetran_job, everything_job
from .schedules import everything_schedule

# Aggregate project definitions
dbt_assets = load_assets_from_modules([dbt_dw])
fivetran_assets = *fivetran_sources.fivetran_assets_google_ads, *fivetran_sources.fivetran_assets_salesforce
all_jobs = [dbt_dw_job, fivetran_job, everything_job]
all_schedules = [everything_schedule]

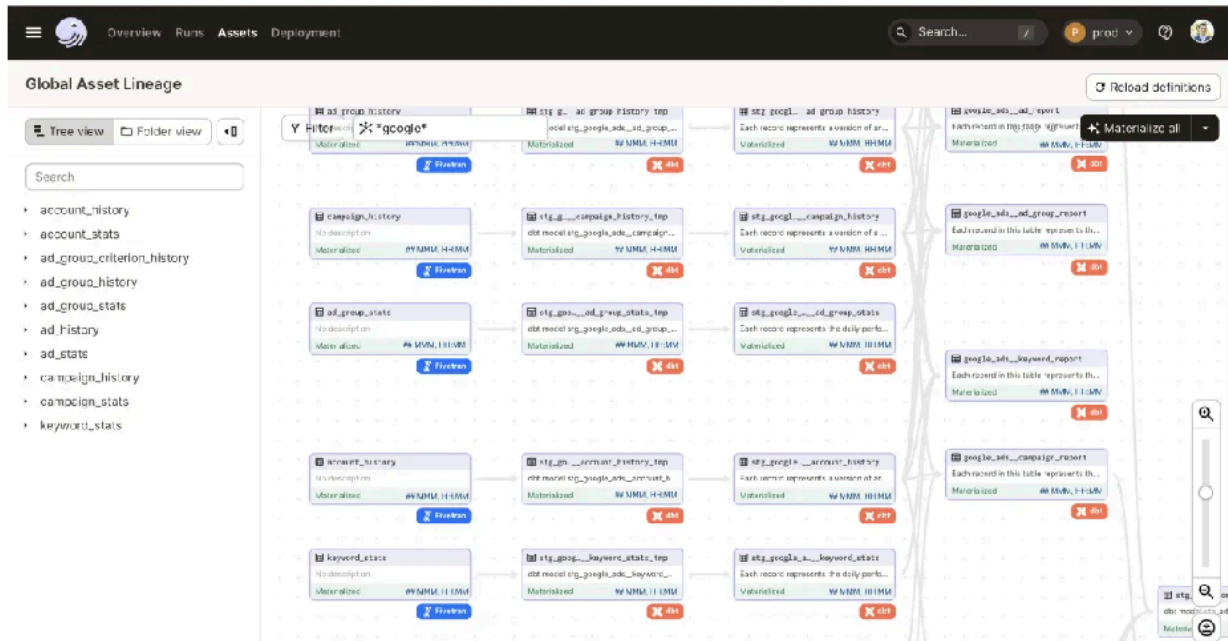
# Definitions object
defs = Definitions(
    assets=[*dbt_assets, *fivetran_assets],
    resources=dbt_project_resource,
    jobs=all_jobs,
    schedules=all_schedules,
)

```

By following these steps, you'll create a well-structured project in Dagster.

## What to expect?

When you orchestrate Fivetran and dbt with Dagster, you open the door to a world of possibilities. Dagster provides a comprehensive overview of your data and analytics infrastructure, breaking down silos and providing insights across teams and systems.



“When something breaks, Dagster enables practitioners to understand where, when, and why it broke”

These assets could be notebooks, app interfaces, or a simple CSV file. As data platform engineers, these are the goods we are bringing to the table, regardless of how clever our engineering is behind the scenes.

By orchestrating Fivetran and dbt with Dagster, you are not only transforming your data infrastructure but also setting the stage for data excellence and agility.

In today’s dynamic landscape of data analytics, the path to success hinges on the efficient orchestration of your data infrastructure. What was once considered a luxury has evolved into a necessity.

It’s not just about managing data — it’s about harnessing its full potential. With the right tools and strategies in place, you can turn raw data into actionable insights that fuel your organisation’s growth and success.

Without an orchestrated approach, organisations find themselves grappling with a tangled web of data sources, inefficient processes, and a lack of collaboration.

The orchestration of Fivetran and dbt with Dagster isn't just a technological innovation; it's a strategic advantage. It empowers you to leverage your data as a strategic asset, making it the driving force behind your organisation's data analytics function.

In conclusion, the choice is clear. Inaction comes with risks, while orchestration offers unparalleled benefits. It's time for organisations to take the bold step towards orchestrating their data infrastructure.



# Coalesce 2023 and the new dbt Cloud CLI

Amir Jaber, October 2023

**C**oalesce 2023 happened a couple of weeks ago in San Diego, London and Sydney and the Rittman Analytics team travelled over to the San Diego event as our annual offsite get-together.

Last year's Coalesce was all about the Semantic Layer, and whilst dbt Labs came back to the topic with the v2, relaunched version of their semantic layer incorporating MetricFlow technology courtesy of the Transform acquisition earlier in the year, the feature that I'm the most excited to try out is the new [dbt cloud cli](#).

```
@tristor - /workspaces/dbt-demo-new (main) $ dbt run -s stg_tch_customers
Sending project to dbt cloud
Successfully created task with id 47b298ee-749-4de9-999-227dd4a8230b
Waiting for response

16:11:43 Running with dbt=1.6.3
16:11:44 Registered adapter: snowflake=1.6.3
16:11:45 Found 63 models, 3 snapshots, , 1 analysis, 68 tests, 2 seeds, 19 sources, 1 exposure,
9 metrics, 647 macros, 0 groups, 6 semantic models
16:11:47 Concurrency: 8 threads (target= 'default')
16:11:47 1 of 1 START sql view model dbt_dguthrie.stg_tpch_customers ..... [RUN]
16:11:48 1 of 1 OK created sql view model dt_dguthrie.st tpch customers ..... [SUCCESS 1 in 1.16s]
16:11:48 Finished running 1 view model in 0 hours 0 minutes and 3.58 seconds (3.58s).
16:11:48 Completed successfully
16:11:48 Done. PASS=1 WARN 0 ERROR=0 SKIP=0 TOTAL=1

Invocation has finished
Artifacts have been downloaded

@tristor - /workspaces/dbt-demo-new (main) $
```

You can now leverage dbt cloud functionality for a more streamlined development workflow without stripping power users of the ergonomics that come with using their IDE of choice (mine: vscode).

The dbt cloud cli is a separate Python package which overwrites dbt-core. They cannot coexist in an environment because they share the dbt namespace when it comes to commands such as dbt run or dbt test.

After installing the package and connecting to your environments in dbt cloud, cli commands will be sent to those cloud instances rather than processing on your computer. Even if the changes to your code are only saved locally and not yet pushed to origin with git, dbt cloud will still process them. That should make it easy to forget that you're not using dbt-core, because your workflow can be exactly the same.

Or, it would be, if it weren't for the fact that dbt cloud processing enables more streamlined access to advanced features. There's some good stuff around not having to tinker with warehouse connection details in your profiles.yml and the new cross-project ref. However I am most excited for the prospect of [automatic deferral](#). From dbt's docs:

“Defer is a powerful feature that allows developers to only build and run and test models they've edited without having to first run and build all the models that come before them (upstream parents). This is powered by using a production manifest for comparison, and dbt will resolve the `{{ ref() }}` function with upstream production artifacts.”

This can save you warehouse spend and time by automatically only running your changed models and reading from production tables / views where possible.

The problem with doing it in dbt-core is that you need access to the production manifest. Making sure that the representative manifest file you have locally is up-to-date with the production one requires consideration and development setup. However, if your queries are being executed in dbt cloud then it handles that process for you. Thus, now you can have easy access to defer without sacrificing your power user ergonomics.

The python package is installed with e.g. `pip3 install dbt` and its installation name here is interesting. The fact that it aligns more closely with the product's brand name (simply, "dbt") implies to me that there might be a deliberate positioning of this package by dbt Labs as the default way people will interface with their dbt projects going forward. "I'm not sure what core refers to but I'm supposed to be installing dbt, so I guess this is the right one".

New users stumbling upon the tool might increasingly only ever know dbt's cloud offering as this workflow matures and docs get rewritten with cloud-focused recommended steps.

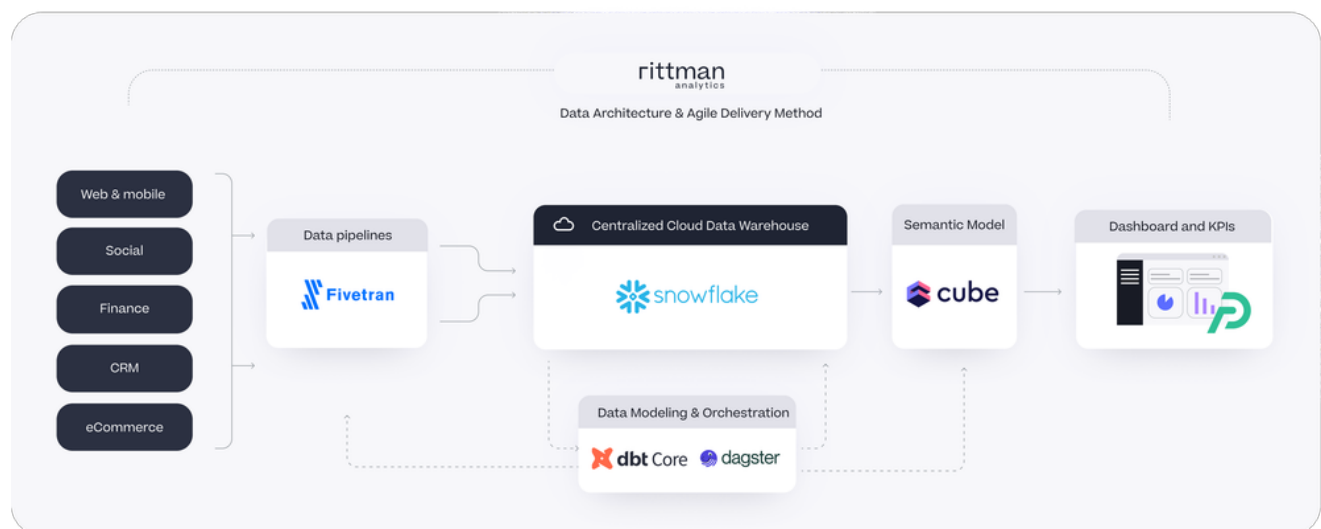
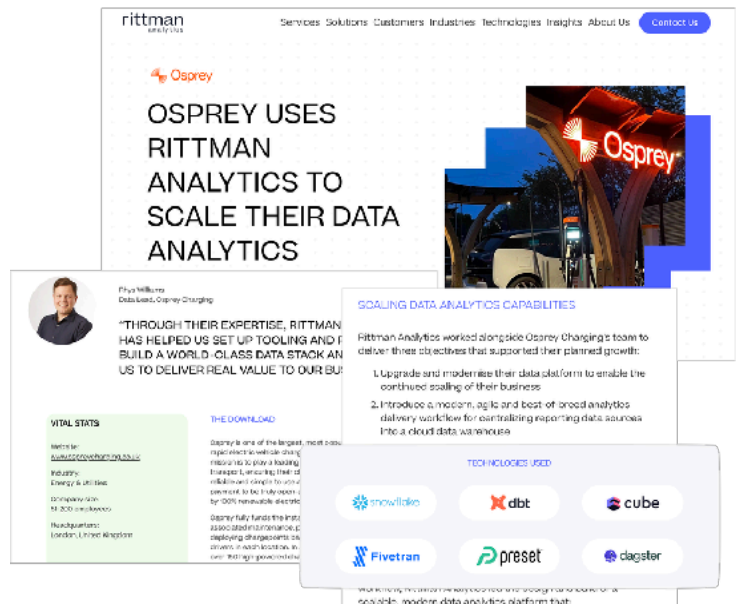
Finally, I do also want to highlight the excellent talk by Niall Woodward and Ian Whitestone from SELECT.dev that's [available now to view on their website](#). It is apparent that they have a deep knowledge of query optimisation and investigation, especially on Snowflake. Their presentation struck a great balance between technical specifics and understandability. The recording of it is definitely a resource I'll refer to a lot.

# How Rittman Analytics Builds Data Stacks for Growth-Stage Businesses using Cube, Dagster and Preset

Mark Rittman, November 2023

One of the [new client case studies we published](#) earlier this month was for our work with [Osprey Charging Networks](#), one of the [most-popular and fastest-growing](#) EV charging networks in the UK and based not far from us in London, England.

As Lewis and I are both EV owners we know the value of a fast, reliable charging network and [with Osprey planning to install over 800 new chargers in 2023](#), we've been working with their team to modernise and scale their data stack and data team to ensure it supports and powers Osprey's growth over



In the past the choice of components for such a data stack would have been limited but as vendors such as Google Cloud (for Looker) and dbt Labs increasingly focus on enterprise, rather than growth businesses and as monolithic BI suites become increasingly disaggregated, we saw an opportunity to recommend a data stack architecture that was both best-of-breed and worked for startups in today's more challenging funding environment.

- Snowflake as the cloud data warehouse, a safe and easy choice
- dbt Core (but not dbt Cloud) for data modeling and transformation
- Dagster Cloud for data orchestration and data asset lineage
- Cube Cloud for the semantic model, caching and role-based access
- Preset Cloud, the commercial version of Superset, for dashboards and reports
- Fivetran for data pipelines and other data ingestion

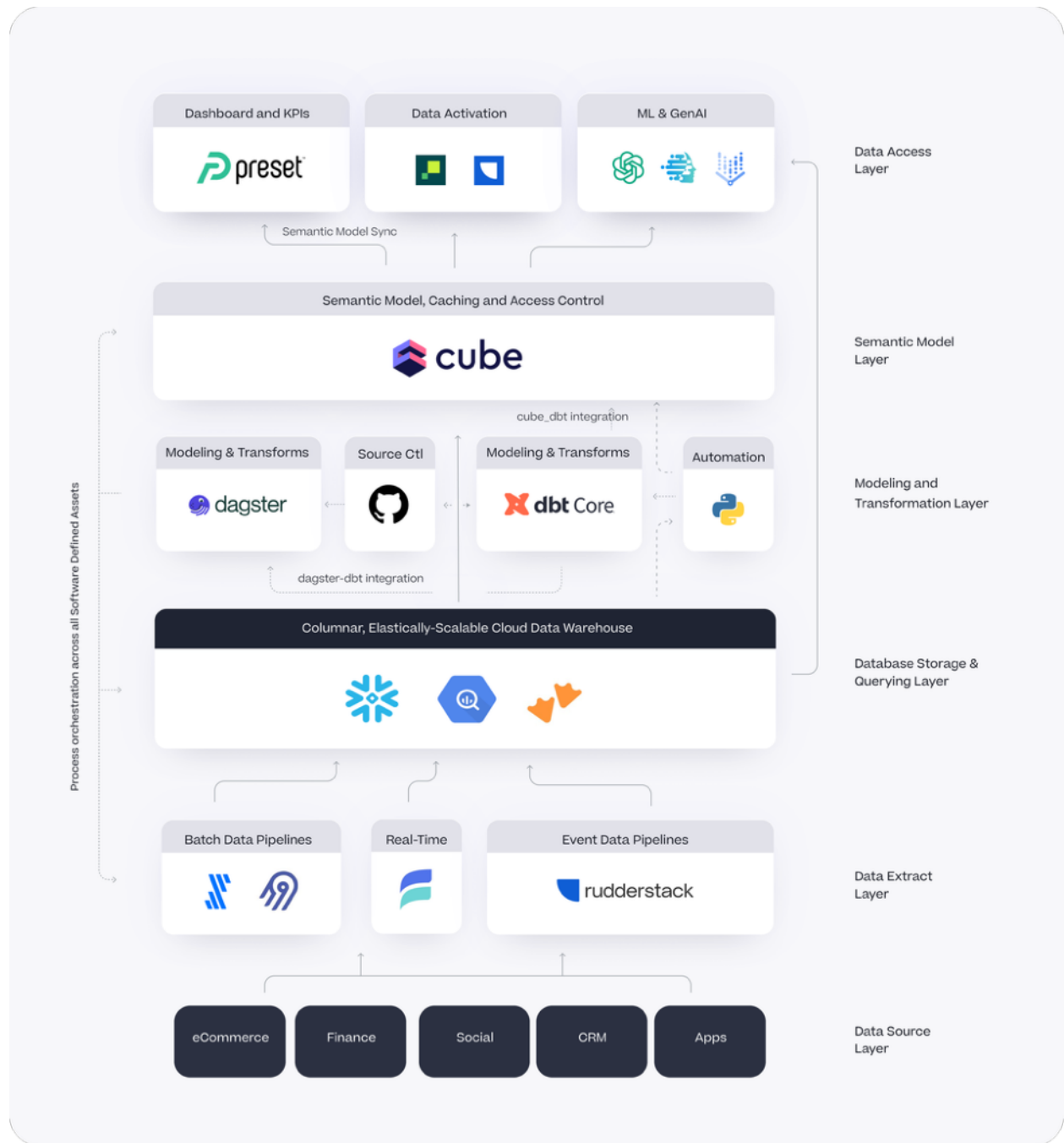
## The Growth-Stage Modern Data Stack

The choice of vendors we recommended for Osprey Charging's data stack is one example of the vendor partnerships and data stack component recommendations we've made this year for growth-stage organisations who we've found have broadly similar needs and goals:

- A desire for a Looker-style dashboarding tool with an integrated business semantic layer
- Data assets and a data architecture that extend beyond just batch SQL transformations updating warehouse tables on an hourly schedule
- A budget for their data initiative that two years ago might well have been north of £250k – £500k but now is likely to be just half or a quarter of that, including spend on getting someone in like ourselves to get the first stage delivered

Generalised beyond just Osprey Charging and with further options such as [Motherduck](#) for the cloud data warehouse, [Rudderstack](#) for adding event collection, [Airbyte](#) for as an alternative to Fivetran for batch data extract and [Estuary](#) for ingesting data in real-time as well as services such as [Rudderstack](#) and [Hightouch](#) for data activation and reverse ETL, the diagram below shows the

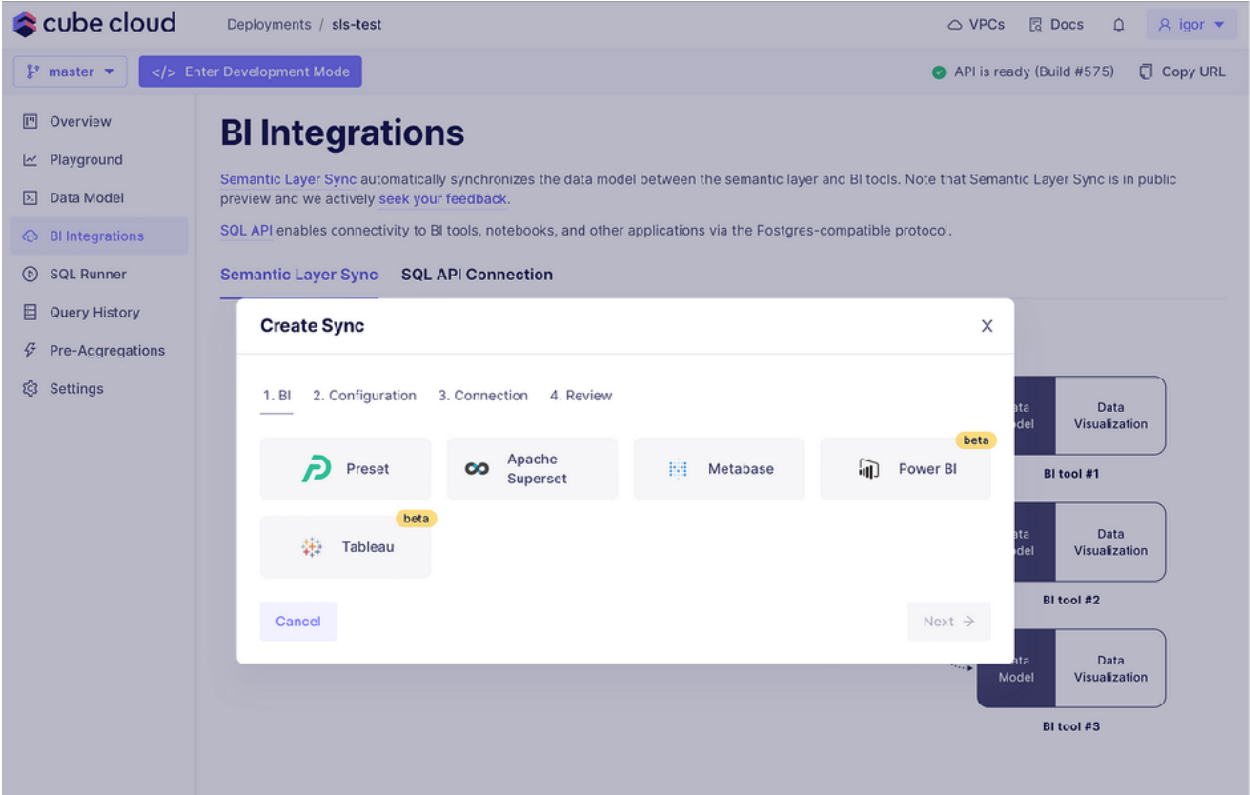
growth-stage data stack layers and services we generally recommend for clients today.



# The Enabler: Cube, Dagster and Preset Integration

There have, of course, always been niche vendors within the modern data stack ecosystem but what makes this type of architecture increasingly viable is the integration between Cube, Dagster and Preset that's now available as features provided by the commercial cloud-hosted versions of those products.

Cube, the disaggregated or “headless BI” semantic layer that also provides role-based access control and tools to pre-aggregate query results in your cloud data warehouse now comes with a [Semantic Layer Sync](#) feature in its cloud version that automatically creates, and then keeps in-sync, dataset definitions in tools such as Preset, Metabase and (coming soon) Power BI.



Another one of our clients, [Breakthrough](#), was an early adopter of Cube's Semantic Layer Sync and found value in [having their BI tool work off the same semantic model as the data products that they embedded Cube into](#) as their original use-case for their technology.

We are just in the initial phase of testing Semantic Layer Sync. So far, having the ability to sync our dimensions and our measures to Preset is going to save us a bunch of time and eliminate any errors in trying to keep them in sync.

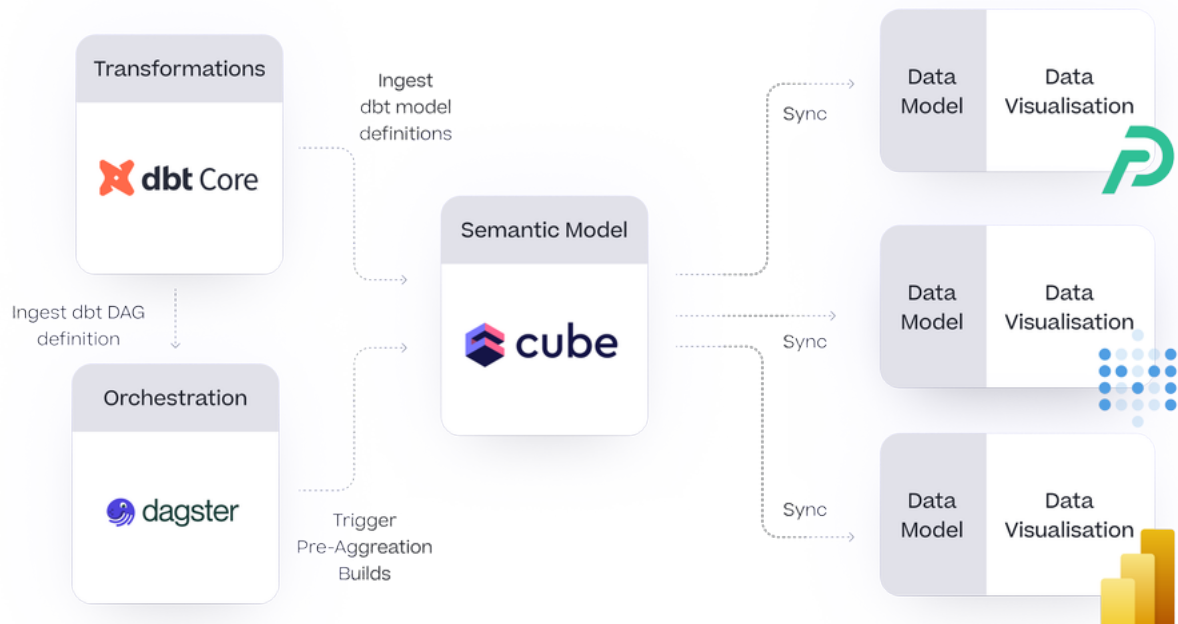
The semantic sync adds a lot of value for use cases like ours, where we need an analytical API that is consumed by an application as well as a BI tool.



**Juan Muñoz**  
Data Strategy and Innovation Lead at Breakthrough

Both Cube and Dagster now have the ability to connect to dbt (via the dbt manifest file or via your dbt package's git repository) to [ingest model definitions](#) and the [entire dbt transformation DAG](#) respectively into their own repositories.

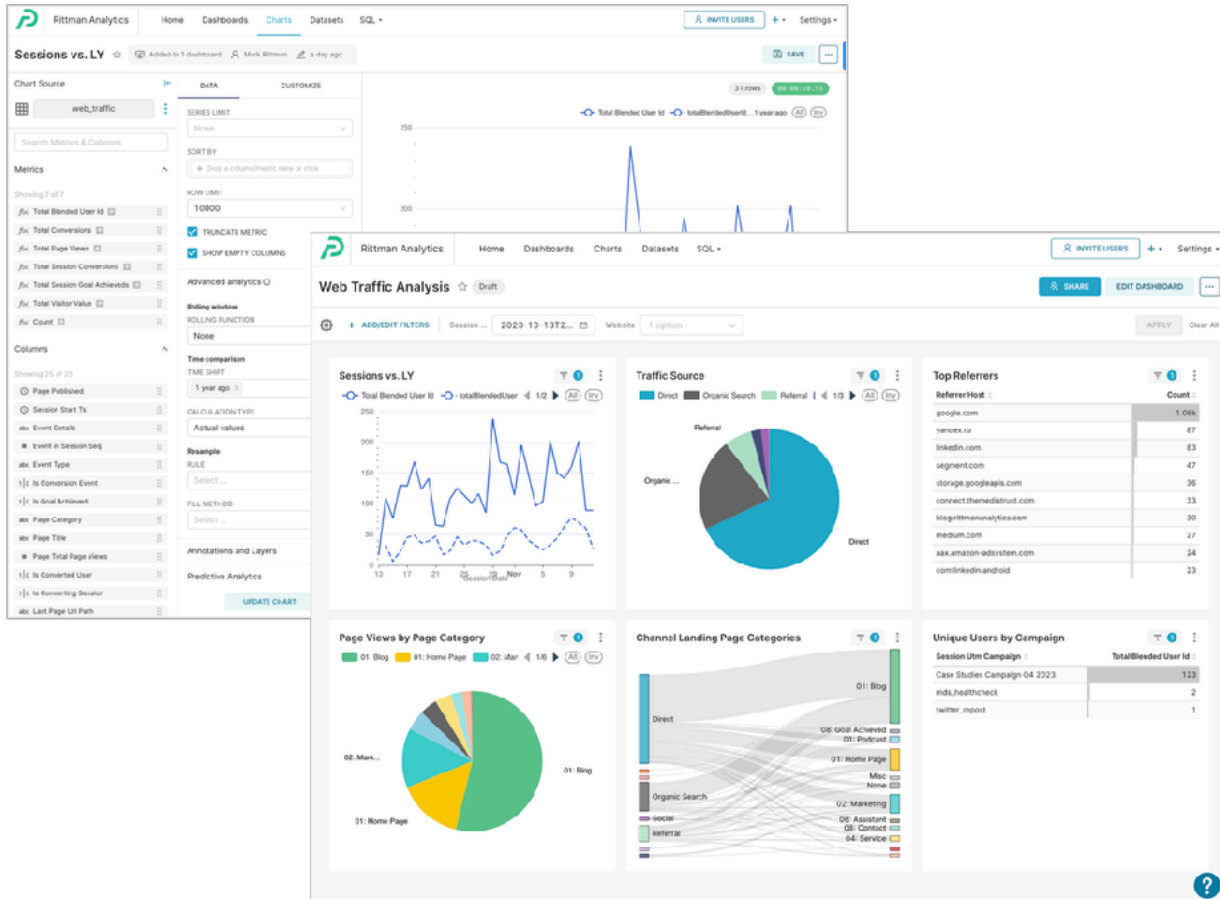
Coupled with [Dagster's ability to trigger builds of Cube pre-aggregations](#) using Cube's Orchestration API gives you a level of component integration as deep and in-fact wider than you had previously with Looker.



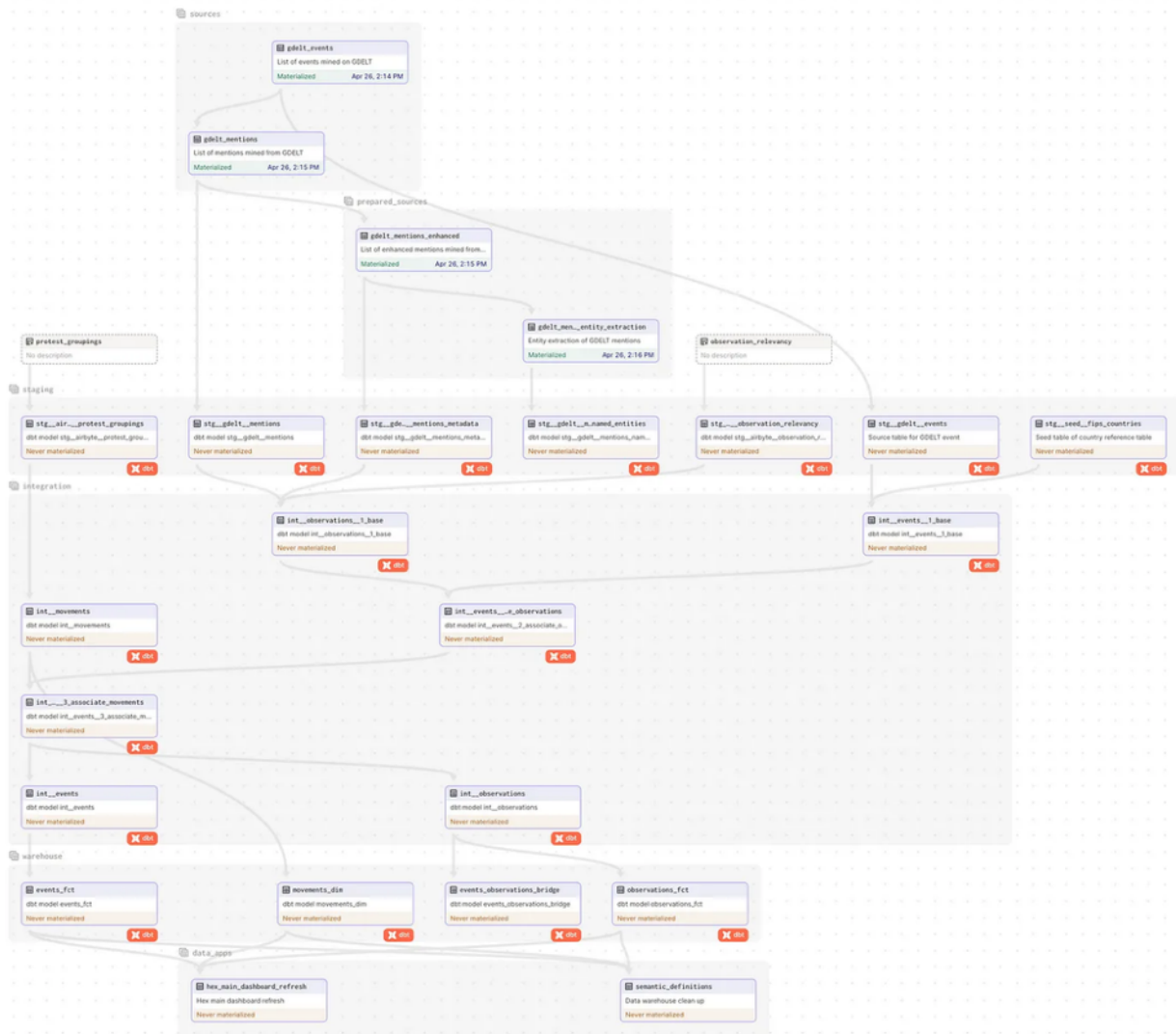
Preset is a great ad-hoc query tool and with this new, closer integration with Cube's semantic layer the user experience is like Tableau connected to a Looker-



style semantic model, something that's possible now with Preset and Cube but still not available with actual Tableau and Looker Modeler at the time of writing despite being announced more than six months ago.



My colleagues Jordan Ilyat and Olivier Dupuis have a number of articles on Dagster including one on [orchestrating Fivetran data extracts using Dagster](#) and [configuring Dagster to refresh data items in your warehouse based on freshness policies](#) rather than a schedule, potentially saving you considerable running costs for your warehouse orchestration now that both Dagster and dbt Cloud have moved to consumption-based pricing.



The real advantage with Dagster comes when you have more than just dbt jobs to orchestrate to refresh data in your data stack, as in-fact most organisations do once you factor-in data extraction and data activation into your overall data architecture and as Olivier does in his Medium article.

## How We Build Growth-Stage Data Stacks

We still love Looker and as a [Google Cloud Partner](#) we're super-excited to see [Looker Modeler](#) when it becomes generally available and in the meantime we're evaluating the new [Open SQL Interface for Looker](#) that makes its semantic model accessible to other tools via its own JDBC driver.

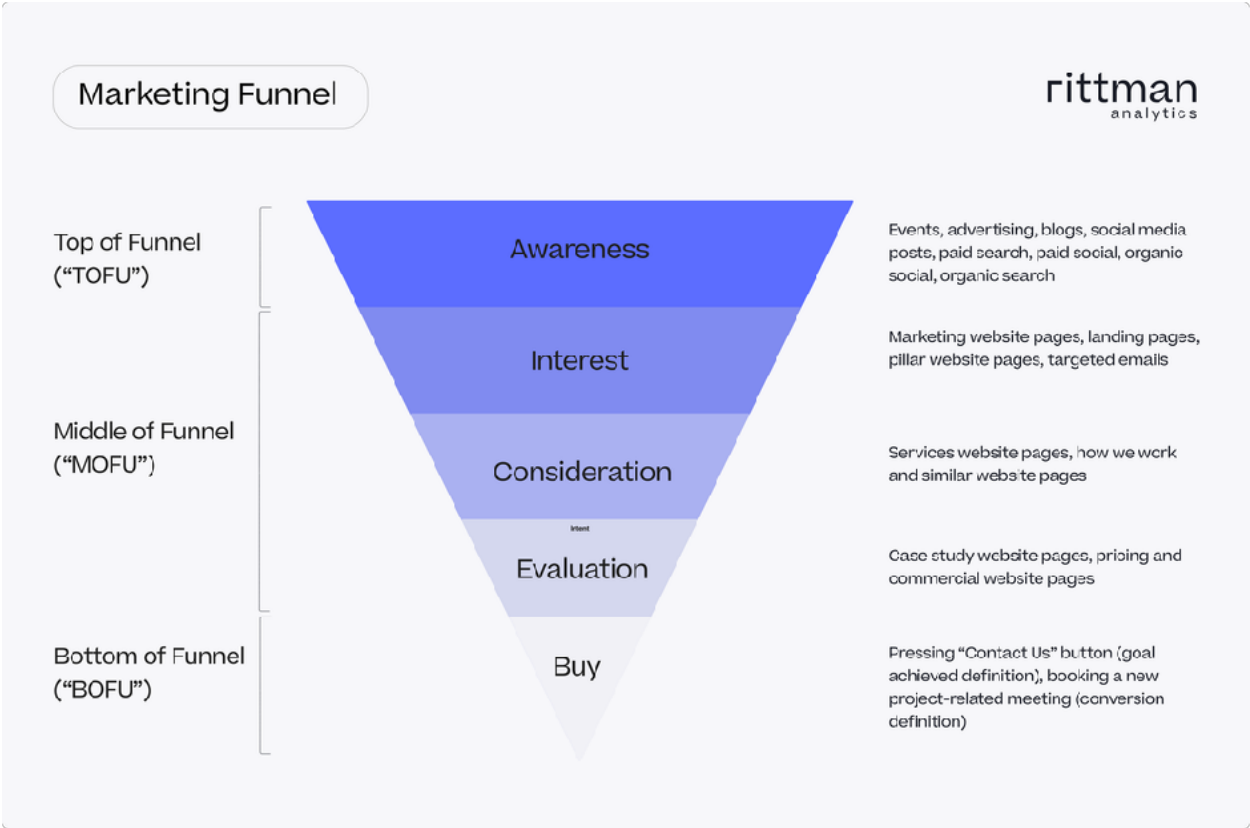
But in the meantime there's now an alternative in the form of Cube, Dagster and Preset that arguably has more features, is just as integrated and can cost a lot less to run – leaving you money in your budget to have us come in and do for your organisation what we did for Osprey Charging Network.

# How Rittman Analytics Does Web Analytics and Marketing Attribution, using Google BigQuery, Looker, dbt and Segment

Mark Rittman, December 2023

**A**s a boutique data analytics consultancy that gets most of its new clients through enquiries from our website, its obviously fairly critical to our growth that potential new clients find and discover our site easily and once they do, stick around and book a discovery call with us to find-out how we can make their analytics vision a reality.

Looking at this in the form of a classic marketing funnel you can divide those marketing activities and the spend they incur into three broad categories:



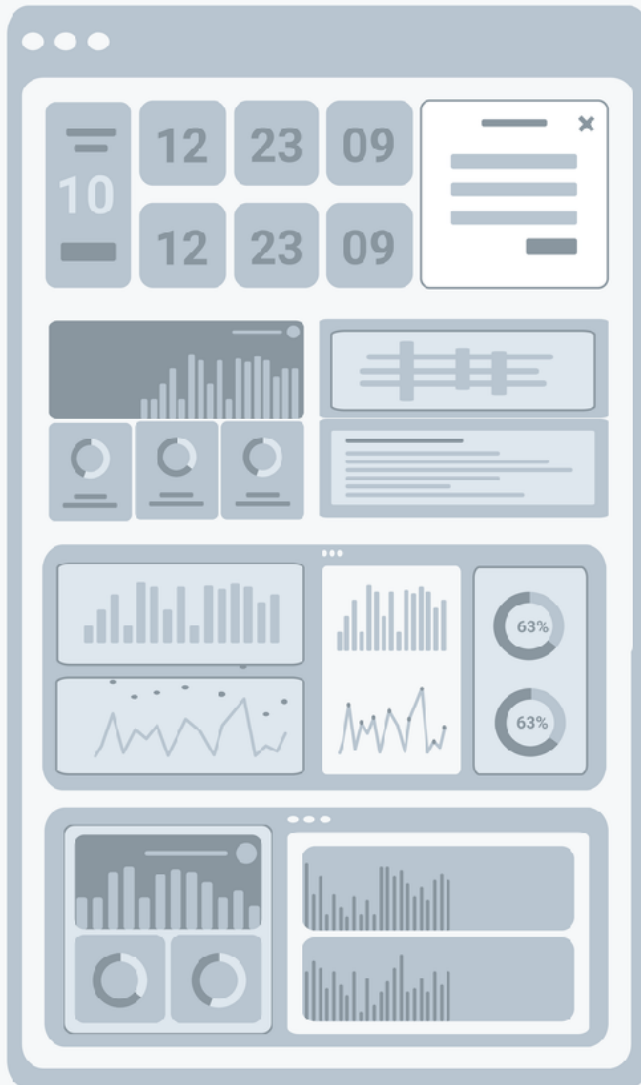
1. “Top of Funnel” activities and touchpoints that attract traffic to our site that will find our content relevant and engaging
2. “Middle of Funnel” website pages and marketing touchpoints that help visitors with an interest in our services find the information they need
3. “Bottom of Funnel” site pages and third-party services that aim to turn that interest into a booked-meeting to find-out how we can help

We therefore wanted to create a web performance dashboard that brought together all of our key web performance metrics in one place, organized by marketing funnel layer and adding enough context in order for those metrics to become actionable.

As we did before with our [business KPIs dashboard](#), the first step was to sit down with stakeholders and come-up with a set of problems to be solved and a process by which solve those problems, expressed in the form of a wireframe dashboard design shown in the diagram below.

Creating such as dashboard that spanned all of our marketing activities, visitor digital touchpoints and data on their transaction history required us to bring data in from a number of different data sources including website clickstream data, lead collection and meeting booking data from third-party services such as [ConvertBox](#) and Calendly, ad network and social media data as well as operational stored in our centralised data warehouse, Google BigQuery.

## Web Performance & Marketing Attribution Dashboard Wireframe



### HEADLINE METRICS & KPIs

# of Sessions, Users, Likes  
Conversion Rate %, Goal Achieved Rate %  
Paid Ad Spend, CPC  
First Conversions, First Goals Achieved  
First Conversion Revenue, First Conversion LTV  
Marketing Funnel Stage #Interactions

### TOP OF FUNNEL METRICS & KPIs

# Sessions by Channel over Time  
Channel Visitor Session Value vs. Channel Avg  
Top 20 Search Keywords by Clicks  
Top 10 Organic Social Post # Likes # Comments  
Paid Social Total Spend, CPC over Time  
Top 20 Search Keywords #Impressions, #Clicks, CTR

### MIDDLE OF FUNNEL METRICS & KPIs

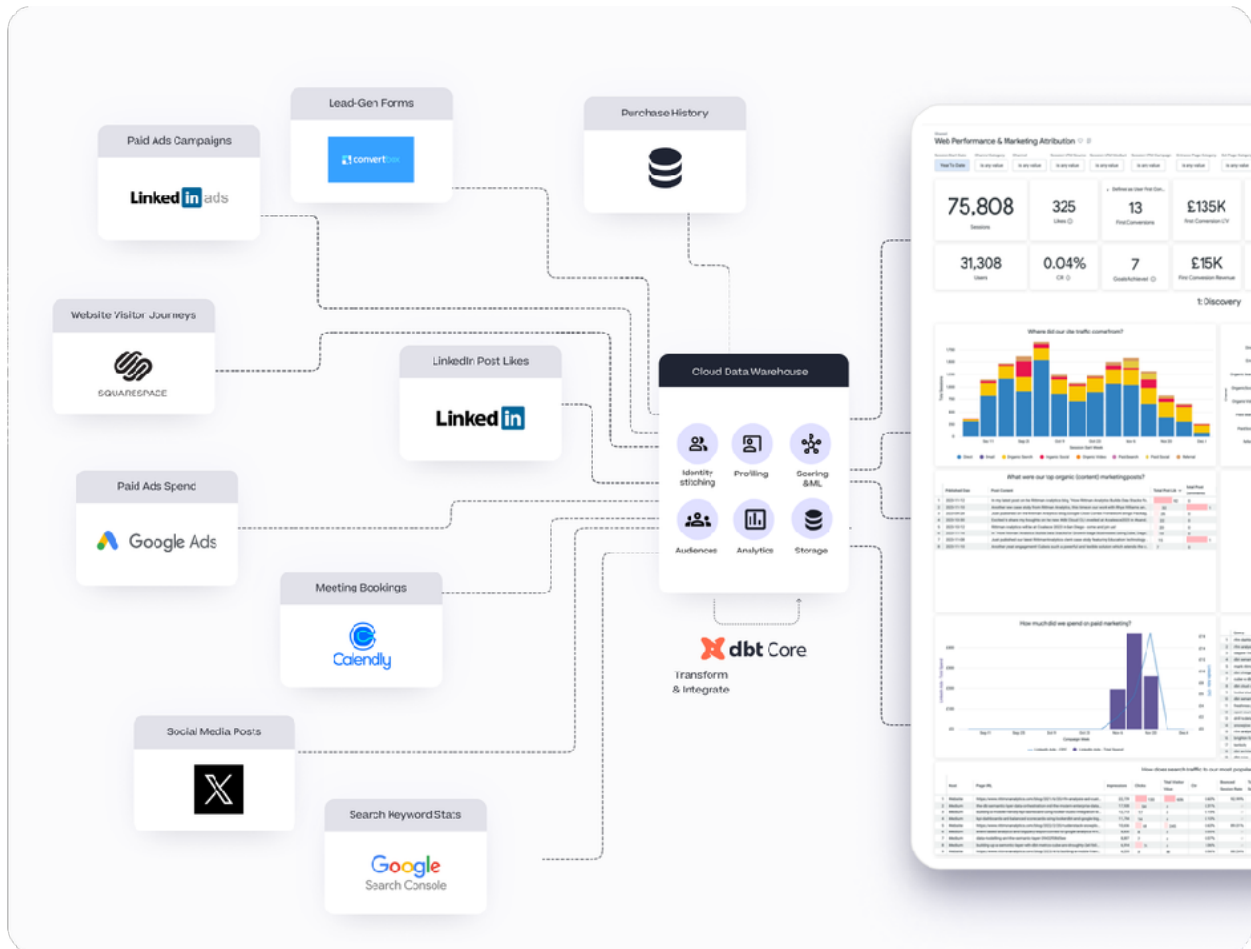
Landing Page Session #PVs by Category  
Organic & Social Media Session #PVs by Category  
Organic & Paid Campaign Session #PVs by Category  
Channel Session #PV Category, CR%, GA% & Value  
Channel Users%, Sessions%, Appraisal PV%, Value %

### BOTTOM OF FUNNEL METRICS & KPIs

Converter with Acquisition Details  
First Non-Direct Click Channel Attribution  
Last Non-Direct Click Channel Attribution  
Linear Click Channel Attribution

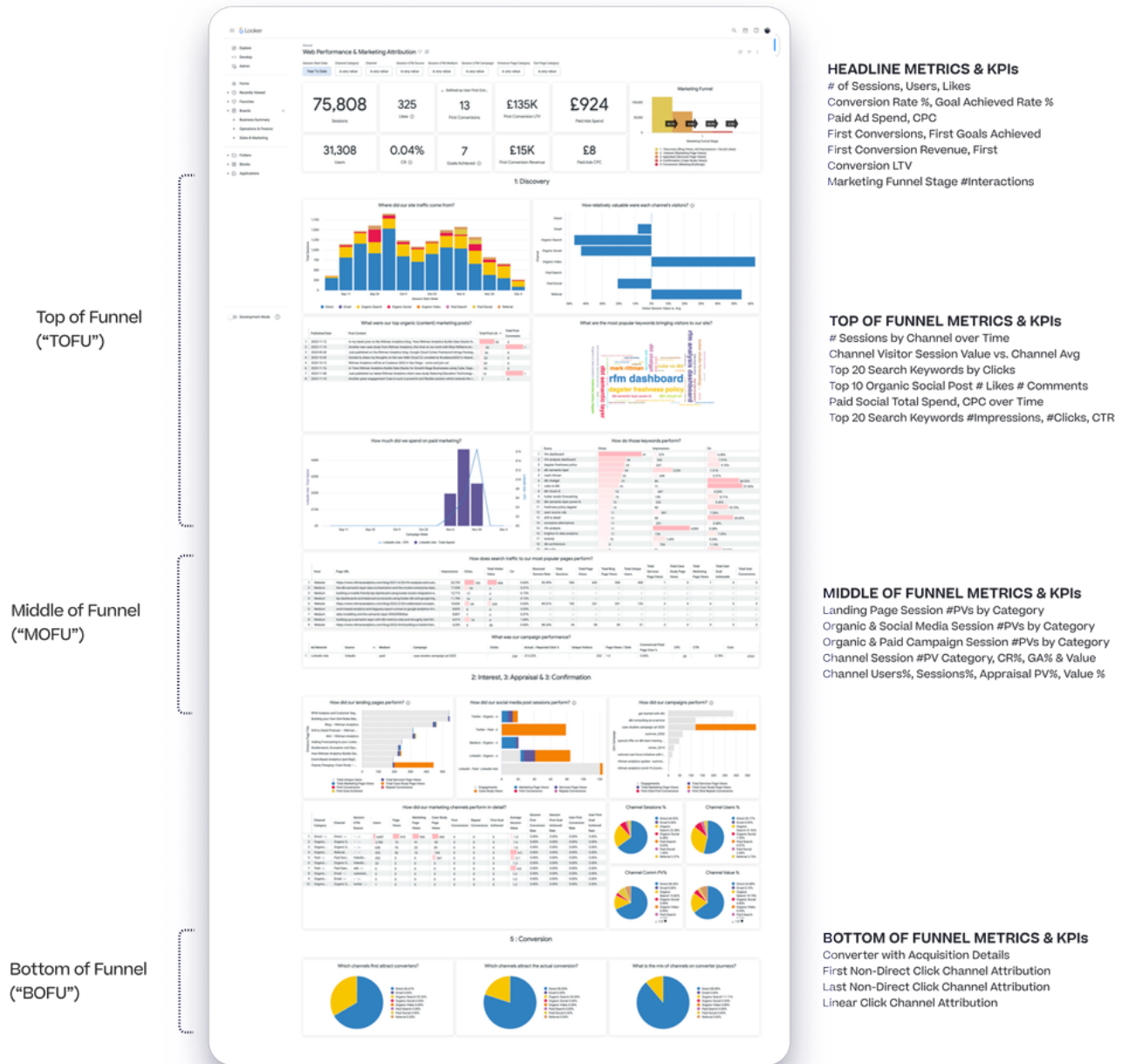
rittman  
analytics

To do this we first landed all of our marketing and customer data sources in [Google BigQuery](#), our cloud data warehouse and then used the open-source [dbt Core](#) toolkit to format, join and then combine all of this data into a single dataset and “version of the truth” ready for building our dashboard in [Looker](#).



We could, theoretically at least have use Looker Studio or even regular Looker to somewhat shortcut this process, to instead create a “single pane of glass” style of integrated dashboard, but this would quickly become unwieldy as the number of data sources increased and would still would leave us with silos of reporting data that could only be analyzed in isolation, not connected to each other.

The Web Performance and Marketing Attribution dashboard is shown in the screenshot below, where we’ve taken the wireframe design and used it as the design template for a Looker dashboard.



The dashboard is organized into four sections, one for headline metrics and three others that correspond to groups of layers in our marketing funnel.

1. Headline Metrics & KPIs : key metrics and indicators that, at-a-glance, give us the state of our marketing activities over the past 30 days
2. Top-of-Funnel Metrics & KPIs : metrics and data visualizations that tell us how we’ve performed at bringing-in high-quality traffic to our site



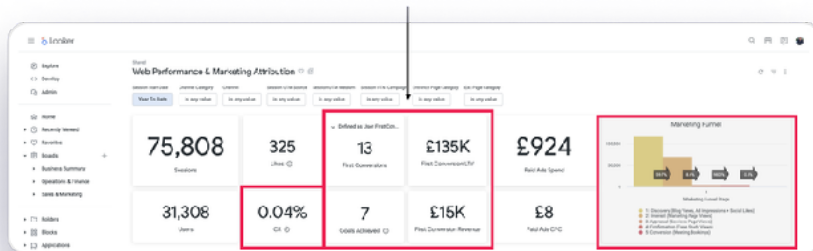
- Middle-of-Funnel Metrics & KPIs : metrics and KPIs that measure how effective we've been at directing and helping that traffic discover relevant content about the services we offer
- Bottom-of-Funnel Metrics & KPIs : metrics and visualizations that show how each of our marketing channels have contributed to conversions (booked meetings) and goals achieved (contact buttons pressed)

## Headline Conversion Metrics and Identity Stitching

For our headline metrics and alongside obvious ones to include such as Sessions, Users, Paid Ad Spend and Cost-per-Click (CPC), we also wanted to highlight performance in bringing-in first meetings with potential new clients as those are the biggest driver of new incremental revenue for our business.

### CONVERSIONS & GOALS ACHIEVED

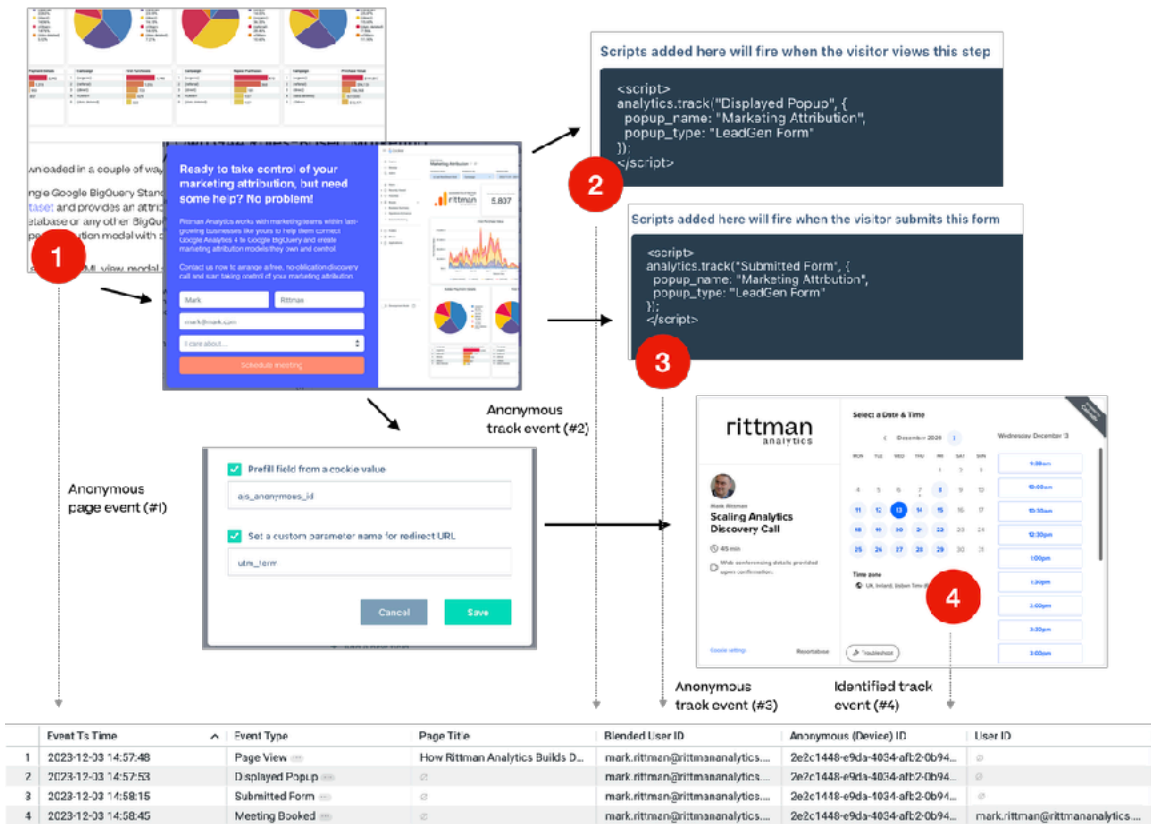
- Number of user first conversions
- User first conversion rate (user first conversions/sessions)
- Number of user first goal achieved
- First Closed Won Deal Value for converting users
- Lifetime Closed Won Deal Value for converting users



### CONVERSION FUNNEL

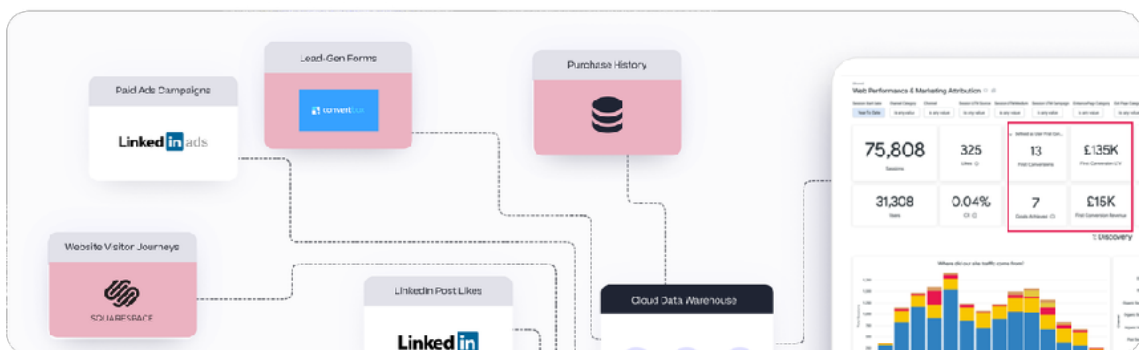
- # of Discovery Interactions (blog page views, paid ad/social impressions + organic social likes)
- # of Interest Interactions (marketing and landing page views)
- # of Appraisal Interactions (services page views)
- # of Confirmation Interactions (customer case study page views)
- # of Conversion Interactions (contact us button presses + meeting bookings)

To emphasise the value of these first meetings we [identity-stitched](#) the anonymous and identified website interactions for each of our converting users, making sure that at all times when we hand-off a user to a third-party service such as Calendly or ConvertBox we always include the device-based anonymousid that Segment assigns to each site visitor.



Doing so ensures that when we send Segment track and identify events from those services back to Segment to record a conversion, we can then connect those events back to the journey that user took anonymously that eventually led to that conversion.

Because we identify our site users using their email address, we can also then connect those users to data from HubSpot, our sales and CRM system that feeds into our operational data warehouse, allowing us to assign first conversion and lifetime value revenue numbers to those converting users albeit with several weeks or months lag between conversion and deal closure.



## Measuring Middle-of-Funnel Performance for acquisition channels and marketing pages

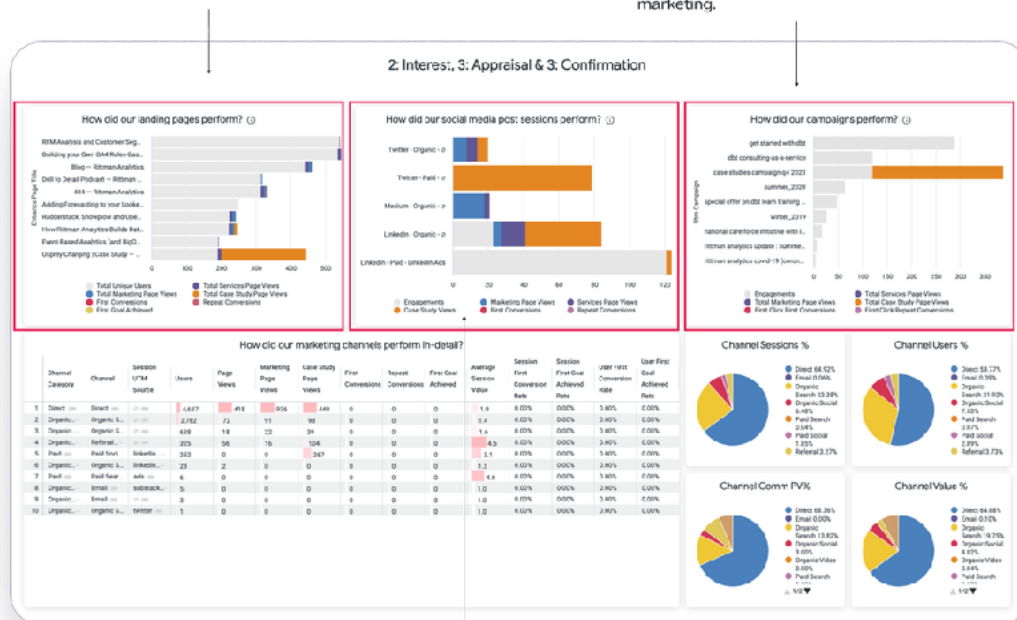
For each of the marketing channels that bring visitors to our website along with blog articles and the marketing, services and case-study pages on our site, our measurement of performance is how effective each were in moving those visitors down through our marketing funnel.

### LANDING PAGE EFFECTIVENESS

For website sessions that started with a visit to one of the blogs, marketing or other pages on our site, how many marketing and other commercial pages did visitors brought in from those campaigns go on to view after finishing that entrance page?

### AD CAMPAIGN EFFECTIVENESS

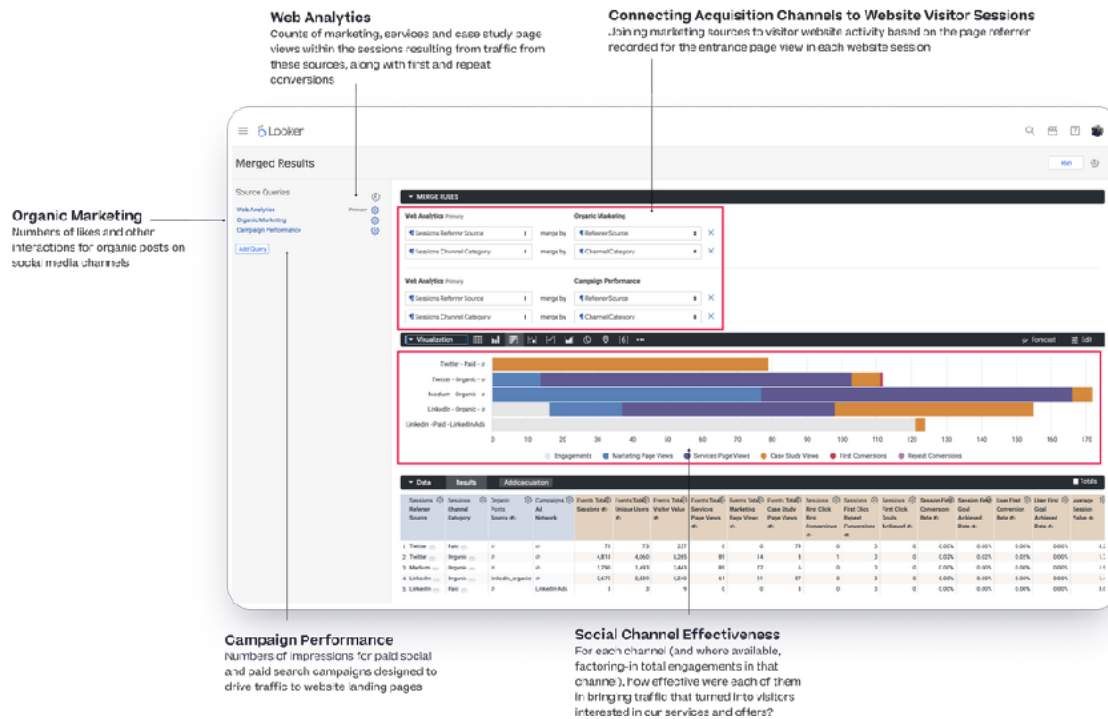
For website sessions resulting from paid advertising and email marketing campaigns, how many marketing and other commercial pages did visitors brought in from those campaigns go on to view - or did they just bounce, an indicator of poorly-targeted marketing.



### SOCIAL MEDIA POSTS

For website sessions that came from traffic that was a result of organic and paid activity on social media channels, how many marketing, case study and services pages did each channel's visitors go on to view along with conversions within those sessions.

Data visualisations such as these that connect-together activity at one stage in the marketing funnel – interacting with a post we published on LinkedIn, liking an X tweet or clicking on a paid social promotion, for example – with subsequent session page view data from the website sessions resulting from those interactions – are easy to set-up in Looker using the Merge Query feature, as you can see from the example below.



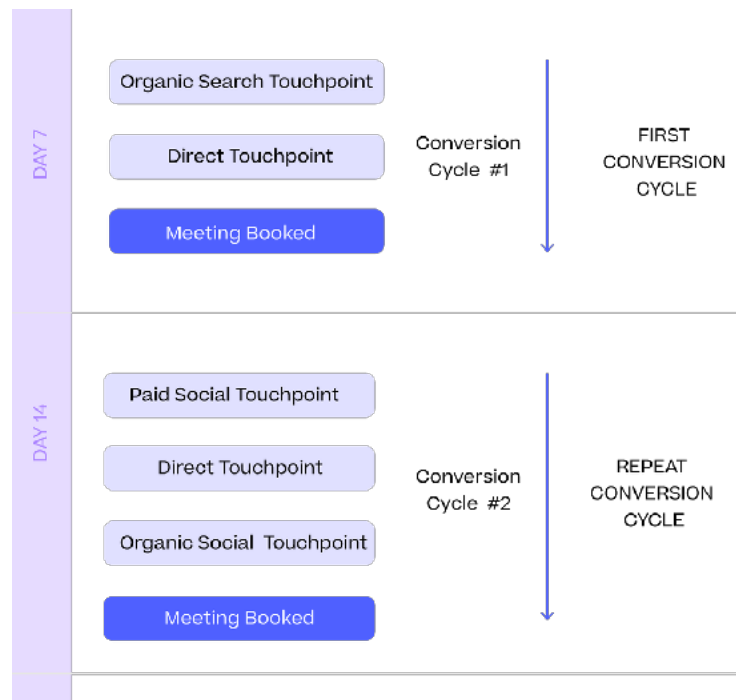
## Attributing Conversions to our Top-, Middle- and Bottom-of-Funnel Marketing Channels

For the final, bottom-of-funnel channel layer of our Web Performance dashboard, reporting on conversions in a meaningful way is surprisingly hard:

- Do we report the total number of conversions (meeting bookings) over a given time period and compare that to the total number of user sessions? Or does session conversion rate (total sessions containing at least one conversion divided by total sessions) or even user conversion rate (number of converting users compared to total users)?
- Are we interested in all conversions by a user, or do we wish to report separately on first conversions (far more important, revenue-wise) and then roll-up any second, third, fourth conversions into a single repeat conversions metric?
- Most importantly though in determining the value each channel delivers for our business, how do we meaningfully attribute that value when last-click attribution (assigning the conversion to the channel the conversion event occurred in) assigns that conversion in most cases to direct.

For users that go on to book meetings its more typical that they arrive by finding us in a Google Search, take a look around and then return a few days later to book a meeting to find out more about our services.

That same user might not be in a position to move ahead right then with their project but returns a week later after being reminded of us through a paid social post on LinkedIn, gets distracted and then finally types our website address into their browser, clicks on the button to book a meeting and converts for a second time.



What we therefore need is a method of attributing conversions to marketing channels that enables us to thereby understand:

- which marketing channels are most effective in bringing-in users that eventually go-on to convert (first non-direct click attribution)
- which marketing channels are most effective in closing that interest and turning it into a conversion (last non-direct click attribution)
- what are the mix of marketing channels that typically feature in a converting visitor journey (even-click or linear attribution)

We do this by leveraging the **multi-touch, multi-cycle marketing attribution dbt package** we've made available on our public Github account (along with a **Looker block version optimized for Google Analytics 4**) and used it, together with the identity-stitching logic outlined earlier in this blog to attribute the value of each users' conversion across each separate conversion cycle, as you can see in the recent example shown below.

**FIRST CONVERSION CYCLE**  
 If we assigned the value of the first conversion to last session in the conversion cycle it would go to the direct channel; if we assigned it to the first non-direct channel it would go to organic search, if we assigned it to the last non-direct channel it would be attributed to referral channel

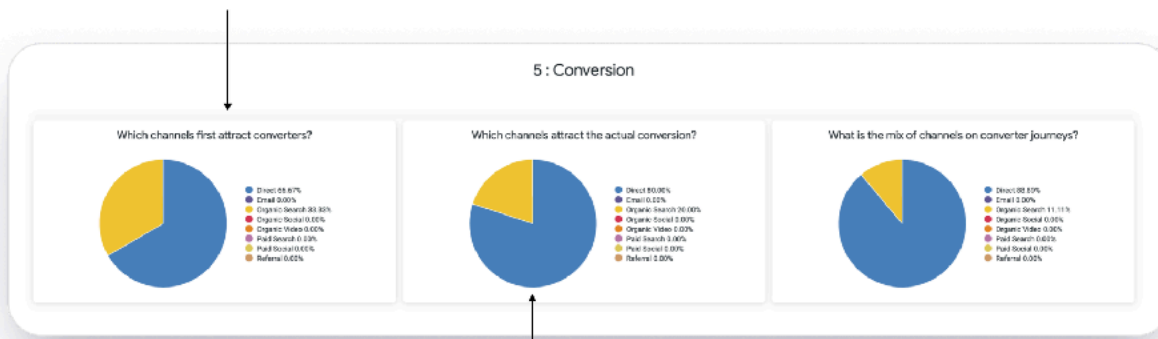
User	Session	Event Timestamp	Event	Website	Is Conversion Session (Yes/No)	Is Goal Achieved Session (Yes/No)	referrer_host	referrer	Channel	UTM Source	UTM Medium	UTM1
anon_4449f9b2-443-916d-73a18b935	1	2028-11-19 17:58:32	Page View	ritmanalytics.com	No	No	google.com	https://www.google.com/	Organic Search			
anon_5646f9b2-443-916d-73a18b935	1	2028-11-19 17:58:58	Clicked Link	ritmanalytics.com	No	No	google.com	https://www.google.com/	Organic Search			
anon_5646f9b2-443-916d-73a18b935	1	2028-11-19 17:58:59	Clicked Link	ritmanalytics.com	No	No	google.com	https://www.google.com/	Organic Search			
anon_10a0e9b2-443-916d-73a18b935	1	2028-11-19 17:59:00	Page View	ritmanalytics.com	No	No	ritmanalytics.com	https://www.ritmanalytics.com/	Organic Search			
anon_4449f9b2-443-916d-73a18b935	1	2028-11-19 17:59:33	Pressed Button	ritmanalytics.com	No	No	ritmanalytics.com	https://www.ritmanalytics.com/	Organic Search			
anon_5646f9b2-443-916d-73a18b935	1	2028-11-19 17:59:32	Clicked Link	ritmanalytics.com	No	No	ritmanalytics.com	https://www.ritmanalytics.com/	Organic Search			
anon_10a0e9b2-443-916d-73a18b935	1	2028-11-19 17:59:32	Clicked Link	ritmanalytics.com	No	No	ritmanalytics.com	https://www.ritmanalytics.com/	Organic Search			
anon_5646f9b2-443-916d-73a18b935	1	2028-11-19 18:00:04	Page View	ritmanalytics.com	No	No	calendly.com	https://calendly.com/	Organic Search			
anon_5646f9b2-443-916d-73a18b935	2	2028-11-19 18:31:21	Clicked Link	ritmanalytics.com	No	No	calendly.com	https://calendly.com/	Referral			
anon_5646f9b2-443-916d-73a18b935	2	2028-11-19 18:31:22	Clicked Link	ritmanalytics.com	No	No	calendly.com	https://calendly.com/	Referral			
anon_5646f9b2-443-916d-73a18b935	2	2028-11-19 18:31:22	Page View	ritmanalytics.com	No	No	ritmanalytics.com	https://www.ritmanalytics.com/assistants	Referral			
anon_5646f9b2-443-916d-73a18b935	3	2028-11-19 18:31:15	Viewing Document	calendly.com	Yes	Yes	ritmanalytics.com	https://www.ritmanalytics.com/	Direct	ritmanalytics		corfa
anon_5646f9b2-443-916d-73a18b935	2	2028-11-19 18:41:07	Clicked Link	ritmanalytics.com	No	No	ritmanalytics.com	https://www.ritmanalytics.com/	Direct			
anon_5646f9b2-443-916d-73a18b935	2	2028-11-19 18:41:09	Clicked Link	ritmanalytics.com	No	No	ritmanalytics.com	https://www.ritmanalytics.com/	Direct			
anon_4449f9b2-443-916d-73a18b935	2	2028-11-19 18:41:10	Page View	ritmanalytics.com	No	No	ritmanalytics.com	https://www.ritmanalytics.com/efranchise	Referral			
anon_10a0e9b2-443-916d-73a18b935	4	2028-11-20 19:11:31	Page View	ritmanalytics.com	No	No	google.com	https://www.google.com/	Organic Search			
anon_10a0e9b2-443-916d-73a18b935	4	2028-11-20 19:11:34	Clicked Link	ritmanalytics.com	No	No	google.com	https://www.google.com/	Organic Search			
anon_10a0e9b2-443-916d-73a18b935	5	2028-11-20 19:20:15	Viewing Document	calendly.com	Yes	Yes	ritmanalytics.com	https://www.ritmanalytics.com/	Direct	ritmanalytics		corfa

**SECOND CONVERSION CYCLE**  
 First non-direct click attribution credits referral (or maybe organic search?) for bringing this potentially-converting visitor back to our website and for closing the conversion, if we use last non-direct click attribution

We then use the first non-direct click, last non-direct click and even (linear)-click attribution model data to provide visualizations for the final marketing funnel layer of our dashboard.

**TOP-OF-FUNNEL CHANNEL EFFECTIVENESS**

While most of the visitors who go-on to convert either type our website URL directly into their browser or more likely, first visited our site beyond the expiry date of our site cookies, a significant share actually arrived through organic search.



**BOTTOM-OF-FUNNEL CHANNEL EFFECTIVENESS**

Looking only at the session in-which conversion happened typically credits the vast majority to direct, but organic search again here has an important role to play making SEO optimization an important part of our conversion optimization strategy.

# 10 Ways Your Modern Data Stack Project Can Fail

A bonus blog post from December 2022 by Mark Rittman

The rise of the modern data stack has made it easier, cheaper and faster-than-ever for businesses to centralize their data. Built on cloud-based services, the modern data stack enables businesses to provide reports and KPI dashboards for business end-users, activating that data to improve the efficiency and effectiveness of their customer interactions.

Over the past few years, we've engaged with users of a modern data stack project who were in the "discovery" or "adoption" phase. These individuals were using services such as Fivetran, Snowflake and Looker to centralize one or two data sources in a cloud data warehouse and to build dashboards for use by "early-adopter" departments and end-users.

Those businesses are now moving through the adoption phase and are experiencing common concerns. These include:

- Data quality
- Data velocity
- Data architecture

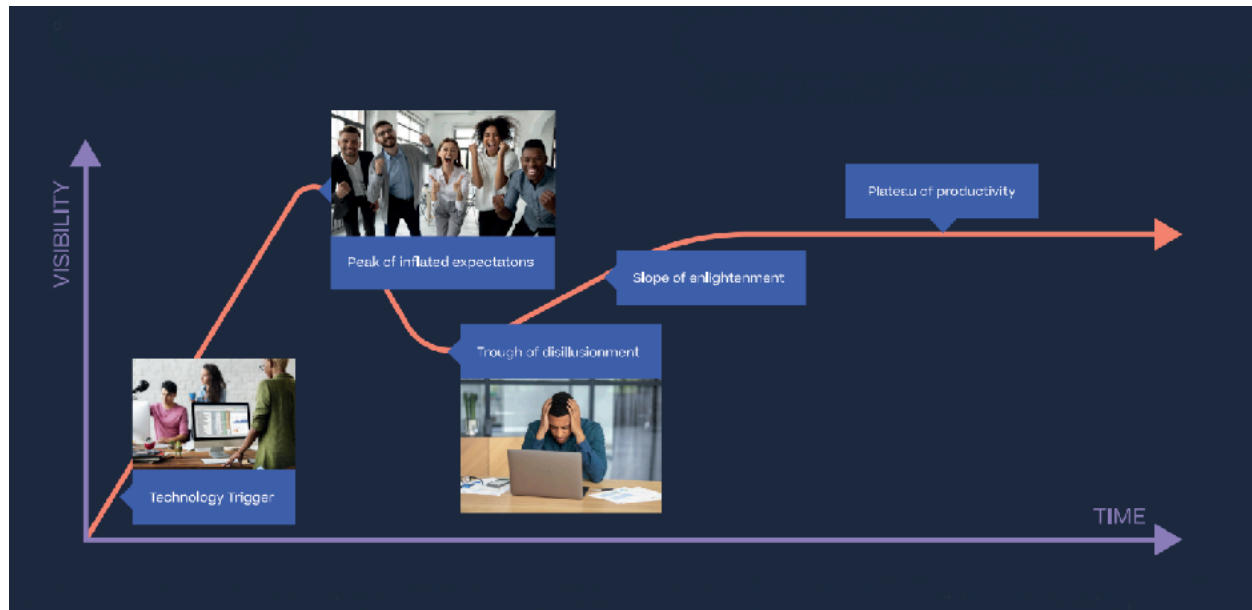
We have also noticed that businesses are struggling to use their data to hit business goals and objectives, due to a lack of clarity and direction of their data stack projects.

With ever-increasing cloud infrastructure costs, the modern data stack could well be entering the "trough of disillusionment" stage in the classic "hype cycle" diagram of technology adoption.

So what are the main ways that your data projects can go wrong and turn your modern data stack into an expensive failure?



Our specialist data team have produced a list of 10 of the most common pitfalls that many businesses are experiencing:



## 1. Unreliable Data that Users Don't Trust

Everyone's got a new dashboard but they still don't trust the numbers. The modern data stack makes it easy to replicate your SaaS data into a warehouse and build dashboards for users, but how do you guarantee that all of the data is correct and keeps your users trust as they use it for decision-making?

The answer is to make testing central to your analytics development workflow, and automate that testing so that it becomes embedded in how your data team works.

Testing of assumptions about new data sources, testing of any new features and functionality provided to end-users and testing of data once it's in production, observing the shape and skew of data that passes all of your validity checks but is clearly implausible in real-life.

## 2. Data Centralisation too Complex to Scale



As new data sources are added to your warehouse, centralising your data becomes increasingly complex.

When none of your systems are the definitive source of customer, product, sales or other key business entities, your design approach of working it out as you go along can quickly leave you with a data model that's complex, confusing and grinding your project to a halt.

Scaling your data centralisation project beyond a trivial amount of sources needs your team to have a set of design patterns and a layered warehouse design such as the [one we use on our client projects](#) that:

- combines identity across multiple systems when none of those systems are the definitive source of that data
- deduplicates multiple sources of customer data even from within the same source system
- keeps the velocity and agility of project delivery consistent even as complexity increases
- and most importantly, makes sure that numbers are trusted, tested and actually add up properly

### 3. Pace of Delivery too Slow for the Business

A new data stack is of little value when it is unable to provide the data you need – when you need it. And, as your business scales, that demand for more data will only grow stronger.

The pace of delivery for your data team needs to match the velocity of your business otherwise it'll quickly become irrelevant, and you're back to the bad old days of decisions being taken by the “loudest person in the room”.

Addressing this issue of project velocity requires three things to be in place:

1. Data team leadership with experience and an ability to communicate clear and achievable goals
2. An analytics delivery approach that delivers “right first time” at a predictable and rapid pace

3. Access to design patterns, toolkits and accelerators that help your data team complete the simple tasks quickly, freeing them up to focus on the more complex problem solving tasks

## 4. Misalignment with Business Goals

Just as fatal to the success of your data stack are projects that deliver but which are misaligned with your business goals.

However hard the data team works and regardless of how innovative their approach, a data team and data stack that are falling short of solving real business problems will struggle to drive their business forward.

To avoid this issue depends upon two things; making sure that your analytics team and data stack are consistently being used to solve complex business problems that deliver near-term, quantifiable and tangible benefits for a key, influential stakeholders.

Its essential that your project has a focus on solving enough of a problem to get something of value delivered, rather than satisfying technical curiosity.

## 5. Lack of Experience with New Technologies

Building a modern data stack requires new tools, technologies, and processes but a central challenge to you is that your new data team lacks experience in this environment.

There just aren't many Heads of Data available for hire who have built a modern data stack, they're expensive and right now, it's unlikely you'd get sign-off for a new full-time hire at this level at all.

A better approach is to bring in a consulting partner such as Rittman Analytics who have worked with and built data stacks for hundreds of venture-funded, fast-growing businesses.

Your consulting partner should also be deeply plugged into the networks of the innovators in the space in order to provide your data team with the context and understanding they need of exactly how these tools work.

At Rittman Analytics, we're typically engaged after A and B-series funding rounds or when a modern data stack project gets the go-ahead in a more established

business, and we usually work directly with the CTO and executive team. We set up analytics infrastructure and processes and build the core warehouse, metrics and reports the business needs.

As our clients' businesses grow, we typically transition to more of a supporting role, helping clients grow their data teams by recruiting, interviewing, and training new team members on best practices and our delivery approach.

## 6. Lack of Planning for Support and User Enablement

Smart data teams enjoy learning and solving new problems; building your new data stack is one of the most interesting and rewarding projects your team can work on right now.

But what do you do when they move on, taking their new skills away with them to another challenge or opportunity? Who do you have left to maintain and extend the data stack that is so critical to the growth of your business?

Consulting partners such as Rittman Analytics address this need by providing ongoing support for data stacks that they or your team built.

This is typically structured through a monthly package of hours and services that maintain and monitor your analytics infrastructure.

Your consulting partners should also be able to help you identify and resolve any issues caused by source data changes, giving you the reassurance that everything is in safe hands.

## 7. Cloud Infrastructure Costs getting Out-of-Control

Back in the old days of on-premises data warehouses, you knew when it was time to start looking at your query costs when you couldn't fit any more servers in the server room.

Today, the only limit to how many Snowflake credits or BigQuery slots your data stack's queries can consume is the limit on your company credit card, and tuning and optimizing the SQL generated by your BI tools and transformation processes

requires specialist skills.

Keeping your cloud infrastructure costs under-control comes down to three things:

1. A development process that considers the cost of ownership of your data pipeline and analytics platform and tests that it runs within acceptable cost parameters
2. A monitoring and altering process that identifies queries and pipeline processes that exceed expected cost limits
3. Experience within the delivery team to tune and optimize the SQL running in your data warehouse

## 8. Data warehouse platforms that don't scale

Of course, the reason nobody wants to be a DBA today is that with Snowflake, BigQuery and managed service Postgres databases, you don't need one as they're largely self-managing.

No partitioning needed for Snowflake, no sizing needed for BigQuery, and you can even run your transactional and ad-hoc query workloads on the same Postgres database.

The challenge comes, however, when your data stack scales to the point where this hands-off, "it-just-works" approach reaches its limit and the only choice is to throw more and more money at the problem ... or accept that this is the limit of how far it will scale.

Or you could work with a partner such as Rittman Analytics who have experience working with database technologies such as Firebolt, designed to work economically and efficiently at-scale.

All database technologies have "sweet-spots" and scenarios that they optimize for, and what you need is a partner on your team who knows when your current approach has reached its limits, what alternatives would provide a better solution and what the new trade-offs of that solution would be.

## 9. Data Pipelines that are always breaking

Your data stack is a success until Engineering releases a new version of your back-end application and your data pipeline breaks.

Something that happens seemingly every few weeks as your engineering and data teams aren't aligned and the resultant outages mean that vital dashboard KPIs aren't updated for hours, even days.

Solving this problem is part process, and part technology. Regular communications and a productive relationship between your data and engineering teams can help identify and resolve potential issues in the future before they impact on your dashboard availability.

Thorough testing and alerting, sandbox and staging environments as well as new concepts such as data contracts can provide smart and thorough solutions as a back-stop.

## 10. You Built it ... but they didn't come.

Perhaps the worst outcome for a modern data stack project is when it's built as planned, but fails to get the level of user adoption you'd hoped for.

A self-service BI project where delivery of new dashboards and reports are still bottlenecked and delivered by a small central team; or the worst fate of all, when your new expensive BI tool is only used to export data to Microsoft Excel!

The reality is that building your warehouse and doing the technical work is only the end of the beginning, and not the beginning of the end; unless you plan now for enabling your end-users to use the platform you've built, and activating your data to innovate and transform your business, a strategy of "built it and they will come" will most likely lead to an expensive failure.

# IS YOUR DATA STACK READY FOR 2024?

Now you've seen some of the most common pitfalls, why not get in touch with us for a deep dive into how well your Modern Data Stack implementation is performing?

Our new Modern Data Stack Healthcheck service gets to the detail of how you're currently using and building your data stack, and compares it to industry best-practices. Together we'll evaluate whether or not your investment has delivered the value it should have, create an action plan and work with your team to make your data stack world-class.



 **Google Cloud Partner**

Rittman Analytics is a Google Cloud Partner that specialises in Data Analytics, Data Management and Artificial Intelligence

Our services transform customer experiences, increase efficiency and create actionable insights, making your data analytics vision a reality.