Folen Data Engineering Case Solution, 29.10.2025

Part 1: Brief Critique and Success Criteria

The brief provided by the team outlines the project requirements at a very high level, giving no detail. It does not mention the timelines, the data availability, or almost any other specific details on the expectations. There are a lot of questions to the team that would need to be asked to clarify the approach, but here are the top 5 most pressing in my opinion:

1. The Multi-touch model is too broad of a requirement - what is the expected logic of the model and would a static rule-based definition be sufficient (like linear or W attribution) or is there a necessity to implement a more advanced data driven scoring?
    - Answering this question makes all the difference in terms of the complexity of data modeling and the rules that would need to be introduced.
2. What are the revenue definitions, i.e. how is marketing-influenced revenue defined? Is it total deal value in HubSpot or is it closed-won deals? What about upsells, renewals and other edge cases that need to be handled? Does "revenue" refer to booked, recognized, or projected revenue?
    - Without the clear definition of terms the expectations cannot be aligned and the modeling of data suffers, as we need to have a clear definition of what defines terms like marketing-influenced revenue, paying customer etc.
3. What are the identifiers that link the lead generation phase to the leads path through the sales pipeline? How do we stitch ad clicks to GA sessions to HubSpot leads (gclid, ga_client_id, hubspot_contact_id, email)? Are offline conversions uploaded to ads platforms?
    - Without a clear way to stitch the customers' ad interaction with their ID in the sales pipeline it is impossible to build a precise model
    - We need to make sure that the GA4 is set up correctly providing detailed data on the source of traffic, and that HubSpot records the user_id from GA4
    - Without those requirements satisfied, we would need to first address them before starting to model the data
4. Do we want to implement a deal level attribution model or contact level attribution model? I.e. Do we want to attribute only the initial deal revenue to the touchpoint, or attribute the whole revenue amount of a client retroactively every time a customer brought in by a certain channel makes a new purchase/deal?
    - Answering this question is crucial for the whole model structure, as contact level attribution requires much more complex modeling but could potentially draw a more complete picture of a customer's CLV.
    - The team needs to decide conceptually on how they want to attribute the deal revenue to touchpoints
5. How much dashboard specific modeling is the team expecting to do on their end? Is the final model expected to be a singular table for direct querying and visualising, or rather a somewhat normalized collection of tables?
    - This of course influences the modeling decisions downstream, and dictates the granularity, complexity of final models

What else would be great to have in a Brief:

- How are the sources of traffic categorised in GA4 data? What kinds of traffic_source and traffic_medium values are associated with LinkedIn ads and google ads traffic?
- The segmentation requirement is vague - what dimensions should be included/prioritised?
    - Without the clear requirements in terms of dimensions it is difficult to prioritise the fields that should be included in the final modelWhat is the expected granularity of the output? Is it user-based, campaign-based or ad-based, etc?
- No mention of data freshness/frequency requirements
    - This dictates the ingestion approach and influences the choice of necessary resources
- What are the user actions that convert a user visiting a website to a potential lead entering the HubSpot Sales pipeline? Is it recorded in the HubSpot pipeline?
- Timing / lookback window not defined. Over what window should touches influence a conversion (30, 90, 365 days)?
- What kind of data is available in each system?
    - Right now the model building operates under heavy assumptions regarding data availability
- Do we want to attribute revenue to all touchpoints or only those that led to the customer acquisition?
    - This question goes back to the marketing-influenced revenue definition

Success criteria (beyond the initial use case)

Make these measurable and testable:

1. **New Channel Integration Ease:** the ability to integrate a new advertising platform into the existing dbt model and generate fully attributed revenue reports within X business days of receiving the source data, demonstrating modularity.
2. **Dynamicity of the built pipeline:** the team can switch or add a new attribution mode**l** and have the new attributed revenue metrics available in the BI tool within X days, proving the model logic is dynamic.
3. **Data Freshness (Update Frequency)**: the final model is updated daily without interruptions maintaining daily data freshness.

Part 2 Data Ingestion

We want to get the data from each of the 4 sources into BigQuery (we assume that this is our warehouse of choice, and then we would use dbt to model the data within BQ).
The assumption here is that we want to just take the raw data and put it as is (or with very little changes) to the BQ dataset for its tables to be used as source models in dbt. This way the whole transformation and modeling would be handled within BQ/dbt, which makes our pipeline an ELT one as opposed to ETL. For all the tools, the initial ingestion should be a full

load, and then once the pipeline is up and running it should be incremental to optimize costs and time.

We assume that the Google Analytics data already contains all raw events data for each user session (including the touchpoints data that the session originates from, either via Google Ads native integration with auto-tagging (GCLID) or using UTMs for LinkedIn Ads (assuming that is correctly set up by Demand Team)).

We then only need to enrich this with the advertisement costs data from LinkedIn Ads and Google Ads.

For each of the ingestions, we opt in to use the native ingestion methods where possible to have as lightweight of a solution as possible, especially given that we need little transformation at the ingestion stage as we opt in to use ELT and handle most transformations in dbt.

Therefore, here is what needs to be ingested:

- LinkedIn Ads
    - Ads costs data, Campaign data, Impressions
    - Ingestion Method -> custom Extraction logic from LinkedIn Marketing API as a Cloud Function with Cloud Scheduler, since LinkedIn does not have the native connector to BigQuery
        - Alternatively, a 3rd party tool like Fivetran can be used, but that would lead to increase in costs
- Google Ads
    - Ads costs data, Campaign data, Impressions
    - Ingestion Method -> BigQuery Data Transfer Service (DTS) with full load at the first ingestion, and switching to incremental daily load afterwards
- Google Analytics
    - The full raw events daily tables for - includes data on touchpoints, user actions on the website, timestamps etc.
    - Ingestion Method -> Google Analytics to BigQuery Connector, with full load at the first ingestion, and switching to incremental daily load afterwards
- HubSpot
    - The Sales pipeline data with Customer/Contact data (data on the customer details to be used as dimensions and GA userID for use as join keys) and Deals data with DealID that gives the details on customer journey through Sales pipeline and, crucially, the Revenue data
    - Ingestion Method -> Hubspot to BigQuery Connector with Application Integration, with full load at the first ingestion, and switching to incremental daily load afterwards

Summing it up:

Source	Data	Ingestion	Process notes

Google Ads	Campaign data, clicks, impressions, costs	BigQuery Data Transfer Service (DTS)	Full load initially, then daily incremental.
LinkedIn Ads	Campaign data, clicks, impressions, costs	Custom extraction via LinkedIn Marketing API (Cloud Function + Cloud Scheduler) or (alternatively) 3rd-party connector like Fivetran	Full load first, then incremental. Lightweight, cost-effective approach.
Google Analytics 4	Raw events per session, including touchpoints, timestamps, traffic sources	GA4 → BigQuery connector -> BigQuery	Full daily tables appended; incremental daily updates. Includes GCLID/UTM to link paid campaigns.
HubSpot	Contacts, leads, deals, revenue, pipeline stage	HubSpot → BigQuery connector -> BigQuery	Full load first, then incremental. Includes GA user ID for joining with GA4.

PART 3 - Data Transformations

These are the assumptions under which the model creation is done:

- GA assigns a unique, anonymous ID to every browser that visits your site - the ga_*clientID*. When a user clicks an Ad -> they land on the website -> GA immediately assigns a ga_*clientID* to their browser -> GA automatically records the source (e.g., Google Ads, LinkedIn Ads, etc.) associated with that ga_*clientID's* session.
- When a user triggers an event that marks him as a lead -> he gets his contact record in HubSpot, that table in HubSpot records associated ga_client_id for the customer.
- The linkedin ads has a correct UTM tagging set up that is recorded in GA4 -> so for linkedin traffic we would have fields like utm_source and utm_campaign converted to also be reflected in traffic_source_source and traffic_source_campaign in GA4 data
- One of the main limitations - the sessions of a user that prior to the session that led to a user becoming HubSpot lead cannot be accounted for, as some of those sessions can be lost due to a different browser for example.

Conceptual issues:

- We want to account for all HubSpot deals a client has, and for each deal and attribute its revenue to relevant touchpoints
- One way to do this is to define a lookback window of N days, and only associate the touchpoints that happened no more than N days prior to the closing of a deal.
- However, the issue with that is that deals for a client that happened close in time to each other can get overlapping touchpoints, which can distort the attribution

- One way to deal with that is to create a ‘deal window’, i.e. a time window between the closing of a previous deal and the closing of a current deal. However, it might so happen that a certain non-first deal would have no touchpoints associated with it (for example if a customer makes a next order directly with a sales rep circumventing the interaction with ads/website), creating a peculiar edge case.
- So, two solutions are possible:
    - A)Either we acknowledge that limitation and mark the deals with no touchpoints/website sessions as ‘offline’ or ‘direct’, and count the revenue from it for AOV but not use it in revenue attribution to touchpoints
        - A.2)A sub-solution for this is to identify such ‘offline’ deals and add their revenues to the previous deal and allocate this ‘bulk deal revenue’ to the touchpoints before
    - B) Or we can implement a time-decay attribution, by a) assigning all of the touchpoints happening before a deal (or with a really large lookback window) to that deal, and then implementing a decay factor, so that touchpoints closer to the deal closing get more revenue attribution, while less recent touchpoints get less attribution

For the purpose of demonstration, the chosen approach is option A.

Source models:

stg_google_ads

- A table on campaign level from Google Ads, with fields like:
    - campaign_id
    - impression counts
    - click counts
    - costs of a campaign etc.
    - add campaign_cost_per_click = campaign_cost / click_count
- Standardized currency definitions

stg_linkedin_ads

- A table on campaign level from LinkedIn Ads, with fields like:
    - campaign_id
    - impression counts
    - click counts
    - costs of a campaign etc.
    - Add campaign_cost_per_click = campaign_cost / click_count
- Standardized currency definitions

stg_ga4_events

- Event level data from Google Analytics partitioned by date (appended the daily raw tables from GA)
- Row per event on a website (page view, form submit, quote fill etc.), with fields like:
    - event date, event type, event name
    - session_id

    - user_id/session_identifier - ga_client_id -> to link sessions and events to HubSpot leads,
    - campaign (to link with cost data from LinkedIn/Google ads)
    - traffic_source and traffic_medium (relating the visits to clicks from a paid platform)
    - Additional dimensions from cookies such as:
        - user_country
        - user_browser
        - user_device
        - etc

stg_hubspot_contacts

- dimension table with data on customers such as:
    - contact_id, email, date_created
    - field identifying what led the customer to become a lead
    - ga_user_id -> to match HubSpot leads with website users

stg_hubspot_deals

- the sales pipeline table with fields like:
    - deal_id, deal_name
    - contact_id (to link with contacts table)
    - deal_stage, deal_stage_entered, deal_closed (to filter on paying customers) -> the definition of a paying customer is still up for a discussion)
    - revenue_amount, currency

Auxiliary tables:

campaign_mapping

- The mapping of campaign names to reconcile campaign coding between google ads and linkedin ads

Joins (intermediate int tables):

- **int_marketing_touchpoints** -> clean and standardize touchpoints in **stg_ga4_events -** change the unit of analysis from event on the website to 'touchpoint'
    a. Create a source_type field where we classify source of traffic (session origin) into 'paid', 'organic' and 'other'
    b. Classify events with the 'touch_category' field, to categorize touches into categories like 'landing', 'conversion', 'engagement' etc.

    c. Organize the touchpoints for each session chronologically by event timestamp

- **int_touchpoint_contact_stitch ->** join **stg_hubspot_contacts** with **int_marketing_touchpoints** on **ga_client_id**
    a. This join lets us tie user sessions/events on the website with the corresponding HubSpot contacts
    b. The resulting table is on one row per event per client(lead) level
    c. The join gets all the session associated with a certain ga_client_id, so that given the use of the same browser the customer journey can actually start before the session that landed a user as a lead in HubSpot
    d. applying filters for touchpoints to include only events from sessions brought about by paid channels (google ads, linkedin ads)

- **int_deals_customer_journey -> j**oin **contacts_with_ga_events** with **stg_hubspot_deals** on **contact_id** -> key logic
    a. The model first treats the **stg_hubspot_deals** source:
        i. Orders the deals for each client chronologically
        ii. For each deal identifies the timestamp that the previous deal for this customer closed, essentially creating ‘deal window’ for each deal
        iii. Filters the deals to leave only closed won deals that have recorded revenue
    b. Actual joining of **contacts_with_ga_events** with **stg_hubspot_deals** on **contact_id**
        i. The join is conditional, making sure that only the touchpoints relevant to each deal are matched to that dea
        ii. Left join chosen to preserve the deals that have no paid based touchpoints and treat them separately later
        iii. Touchpoint/event needs to occur within that ‘deal window’, plus account for 90 day lookback window (the window size can be adjusted
        iv. This logic assumes that we want to attribute the revenue from a deal only to touchpoints directly preceding the deal closing, with no overlap (surely a simplistic approach, chosen for demonstration purposes, please read the ‘Conceptual Issues’ breakdown at the beginning of this chapter)
    c. This is the cornerstone model, which already allows for the attribution model to be built
- **int_full_joined ->** join **int_deals_customer_journey** with **stg_linkedin_ads** and **stg_google_ads** on **campaign_id**
    a. Combines (UNION ALL) the campaigns costs data from both google ads and linkedin ads
    b. Joins based on ‘placeholder’ logic based on campaign names
- **fct_revenue_attribution**
    a. Does the actual revenue attribution -> for now a simplistic linear attribution

        i. Caclulates the number of touches per deal_id, and divides the revenue_amount of that deal by the number of touches
    b. Additionally calculates the total AOV (including deals that were not directly associated with paid ads based sessions)
    c. Adds a flag ‘is_offline’ for those offline based deals
- **fct_kpis**
    a. Takes a step further to give an example query based on **fct_revenue_attribution** table, calculating ROAS, Cost per Acquisition, and AOV attributed to paid campaigns

Here is the link to an open github repository with the code:

https://github.com/KirillKovalchuk02/foleon_case_repo/