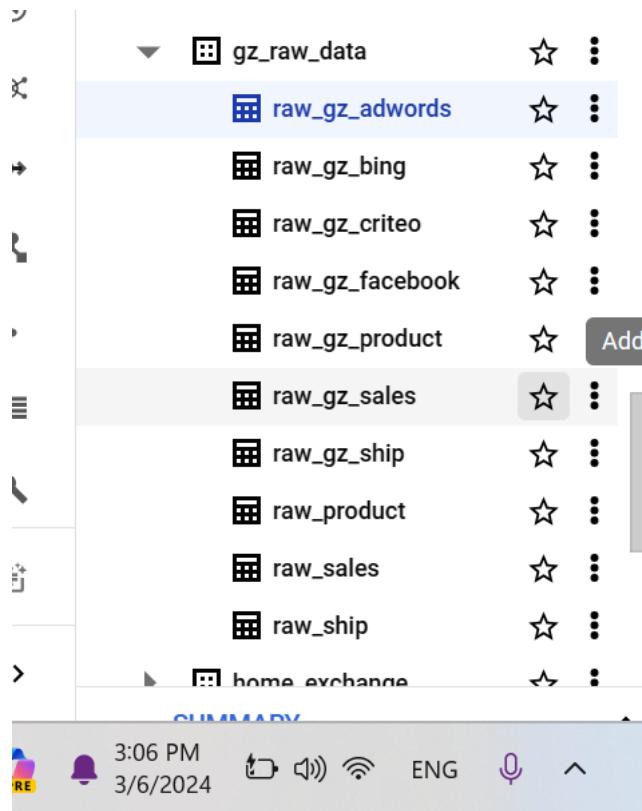


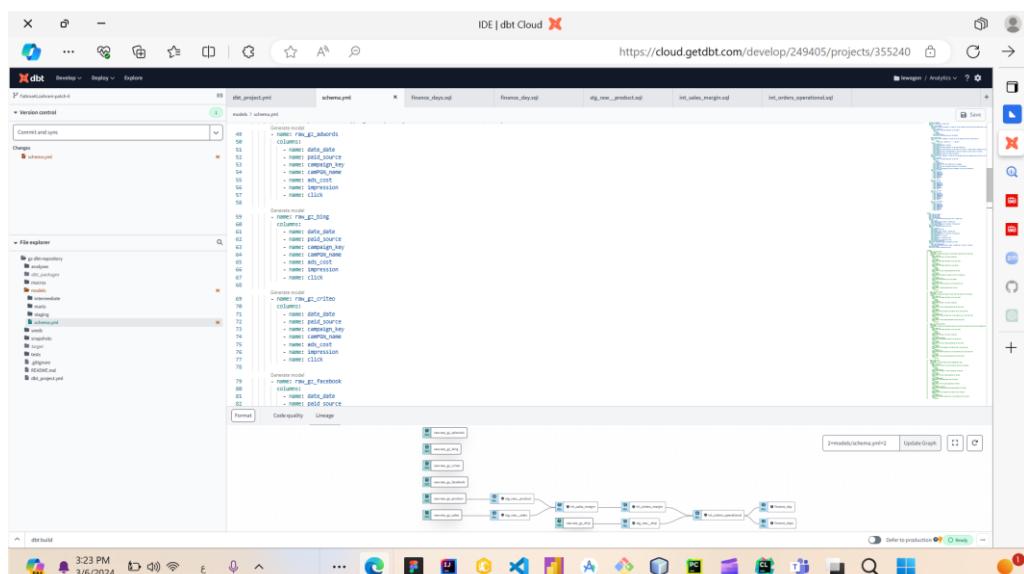
## 1 - Add the Source

1. Copy the following tables in your `gz_raw_data` dataset into your own BigQuery project:
  - o [raw\\_gz\\_adwords](#)
  - o [raw\\_gz\\_bing](#)
  - o [raw\\_gz\\_criteo](#)
  - o [raw\\_gz\\_facebook](#)

*Help - [Google documentation](#) to copy a table*



2. Add the source to your `schema.yml` in dbt cloud



### 3. Generate staging models with the automatic process

The screenshot shows the dbt Cloud IDE interface. On the left, the 'File explorer' sidebar lists various files and folders, including 'schema.yml'. The main area displays the contents of 'schema.yml' and a detailed lineage graph. The lineage graph illustrates the flow of data from raw sources like 'raw\_gz\_adwords' and 'raw\_gz\_bing' through various intermediate and staging models such as 'stg\_raw\_product', 'int\_sales\_margin', and 'int\_orders\_operational' to final operational models like 'int\_order\_operational' and 'finance\_day'. A status bar at the bottom indicates 'Defer to production' and 'Ready'.

### 4. Clean the staging models

- rename `campaing_name` correctly
- cast `ads_cost` from string to float64

The screenshot shows the dbt Cloud IDE interface with a code editor open. The code editor contains a PQL (PostgreSQL Query Language) query for renaming a column and casting a type. The query is as follows:

```
1 with
2   source as (
3     select * from {{ source('raw', 'raw_gz_adwords') }}
4   ),
5   renamed as (
6     select
7       data_date,
8       paid_id,
9       campaign_key,
10      campaign_name AS camping_name,
11      cast(ads_cost AS float64),
12      impressions,
13      click
14    from source
15  )
16  select * from renamed
```

The code editor includes tabs for 'Preview', 'PQL', 'Build', 'Format', 'Results', 'Code quality', and 'Lineage'. The lineage graph below the code editor shows the relationship between 'raw.raw\_gz\_adwords' and 'stg\_raw...raw\_gz\_adwords'.

### 5. Build all your staging sources

The screenshot shows the Google Cloud BigQuery Studio interface. The left sidebar is the 'Explorer' pane, showing a tree view of resources under the project 'dbt\_fatimah', including various models like 'finance\_day', 'int\_sales\_margin', and 'stg\_raw\_product'. The main area is titled 'Welcome to BigQuery Studio!' and features a 'CREATE SQL QUERY' button. Below it, there's a 'Recently accessed' section with four recent queries: 'raw\_gz\_bing', 'raw\_gz\_criteo', 'raw\_gz\_ad...', and 'raw\_gz\_face...'. At the bottom, there's a 'Try with sample data' section and a link to the 'Google Trends Demo Query'.

## 2 – Create int\_campaigns.sql Model

We want to aggregate all the campaign costs referenced in the different tables into an intermediate campaign model.

1. Make sure all the columns have the same name across the different tables, same for datatype.

Done

2. Use UNION ALL to build one single table from the different staging models.

The screenshot shows the dbt Cloud IDE interface. The top navigation bar includes 'IDE | dbt Cloud', 'Develop', 'Deploy', 'Explore', and a URL 'https://cloud.getdbt.com/dev...'. The main workspace shows a project structure with files like 'dbt\_project.yml', 'schema.yml', 'finance\_days.sql', 'int\_campaigns.sql', etc. The 'File explorer' sidebar lists various dbt components: 'gz-dbt-repository', 'analyses', 'dbt\_packages', 'macros', 'models' (including 'intermediate' and 'staging' subfolders), and 'marts'. The 'int\_campaigns.sql' file is currently selected. The code editor contains the following SQL:

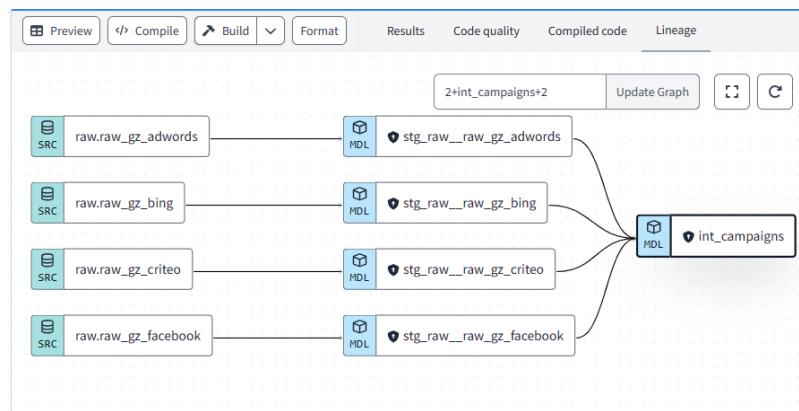
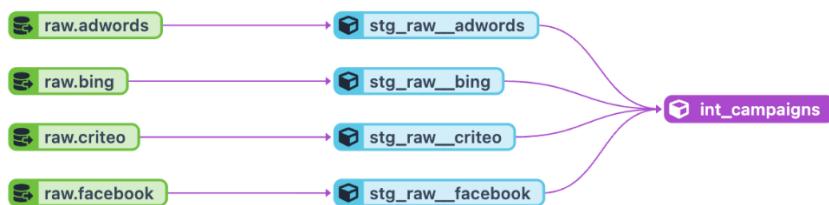
```
1 SELECT *
2 FROM {{ ref('stg_raw__raw_gz_adwords') }}
3 UNION ALL
4 SELECT *
5 FROM {{ ref('stg_raw__raw_gz_bing') }}
6 UNION ALL
7 SELECT *
8 FROM {{ ref('stg_raw__raw_gz_criteo') }}
9 UNION ALL
10 SELECT *
11 FROM {{ ref('stg_raw__raw_gz_facebook') }}
```

Below the code editor is a results table titled '1.9s | Results limited to 500 rows.' It displays five rows of data:

date_date	paid_source	campaign_key	campaign_name	ads_cost	impression
2021-07-31	Facebook	6194131626575	IA - COM [PUR - AC... >	134.0	34940.0
2021-08-03	Facebook	6194131626575	IA - COM [PUR - AC... >	42.0	12238.0
2021-08-02	Facebook	6194131626575	IA - COM [PUR - AC... >	120.0	40864.0
2021-07-28	Facebook	6194131626575	IA - COM [PUR - AC... >	120.0	43848.0

At the bottom of the interface, there are buttons for 'Preview', 'Compile', 'Build', 'Format', 'Results', 'Code quality', 'Compiled code', and 'Lineage'. A 'Defer to production' button is also present.

3. Build your model or `build with upstream` dependencies if you haven't built the staging model yet. The lineage should look like this:



4. Add tests and description to `schema.yml`
  - o Test primary key
  - o Add relevant tests to other columns

```

models > schema.yml
Generate model
50   - name: raw_gz_adwords
51     description: raw_gz_adwords table
52     tests:
53       - unique:
54         column_name: "(campaign_key || '-' || date_date)"
55       - not_null
56     columns:
57       - name: date_date
58         description: the date of purchase
59       - name: paid_source
60         description: The source from which the ad was paid for
61       - name: campaign_key
62         description: The unique identifier for the ad campaign.
63       - name: campGN_name
64         description: The name or title of the ad campaign
65       - name: ads_cost
66         description: The cost incurred for running the ad campaign
67       - name: impression
68         description: The number of times the ad was displayed to users.
69       - name: click
70         description: The number of times users clicked on the ad
71

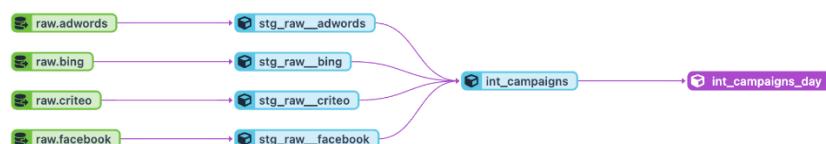
```

## 3 - Create `int_campaigns_day.sql` Model

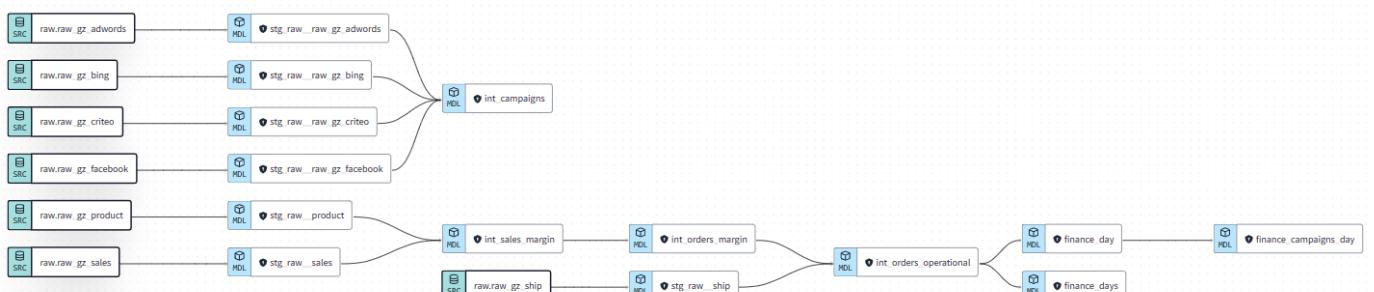
You want to create an aggregated version of `int_campaigns` model for each day.

1. Perform the transformations to create the following table:

date_date	ads_cost	ads_impression	ads_clicks
2021-07-31	3701.0	1345434.0	12552.0
2021-08-03	4273.0	1680171.0	14035.0
2021-08-02	4214.0	1650379.0	13909.0



2. Add tests and description to `schema.yml`



## 4 - Create finance\_campaigns\_day.sql Mart Model

- Join the int\_campaigns\_day model with finance\_days model from the previous lecture.

We want to compute the ads\_margin column requested by marketing and finance team.

$$\diamondsuit \text{ Ads margin} = \text{operational margin} - \text{ads cost}$$

Your preview should look like this:

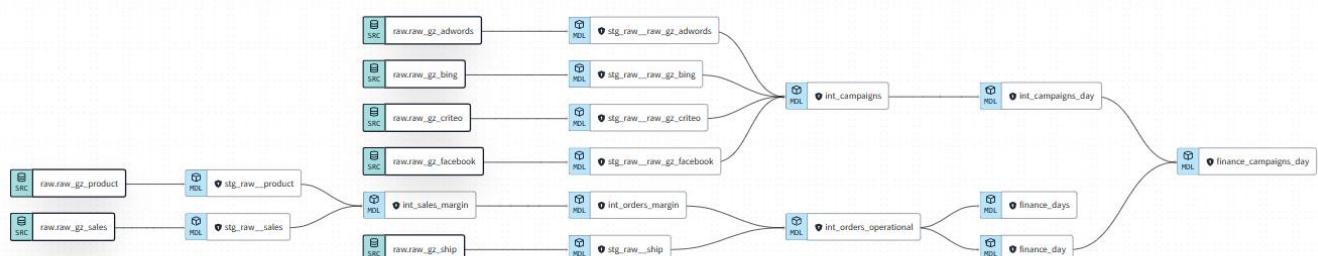
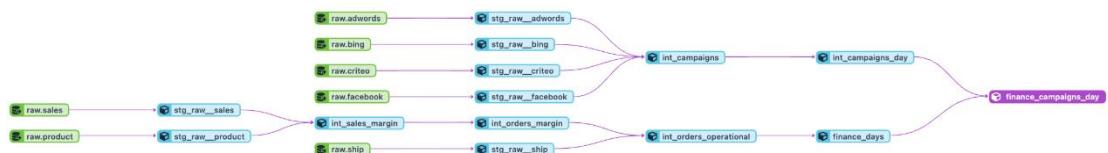
date_date	ads_margin	average_basket	operational_m...	ads_cost
2021-09-30	7794.0	78.4	11287.0	3493.0
2021-09-29	6882.0	70.2	11708.0	4825.0
2021-09-28	7925.0	73.7	13361.0	5436.0

- Add tests and description to schema.yml

```

SELECT
    date_date,
    operational_margin - ads_cost AS ads_margin,
    ROUND(average_basket,2) AS average_basket,
    operational_margin,
    ads_cost,
    impression,
    click,
    quantity,
    revenue,
    purchase_cost,
    margin
FROM {{ ref('int_campaigns_day') }}
FULL OUTER JOIN {{ ref('finance_day') }} ON
    USING(date_date)
ORDER BY date_date DESC
  
```

- Check your lineage.



## 5 - Create finance\_campaigns\_month.sql Mart Model

The finance team ask you to deliver the same table but aggregated for each month.

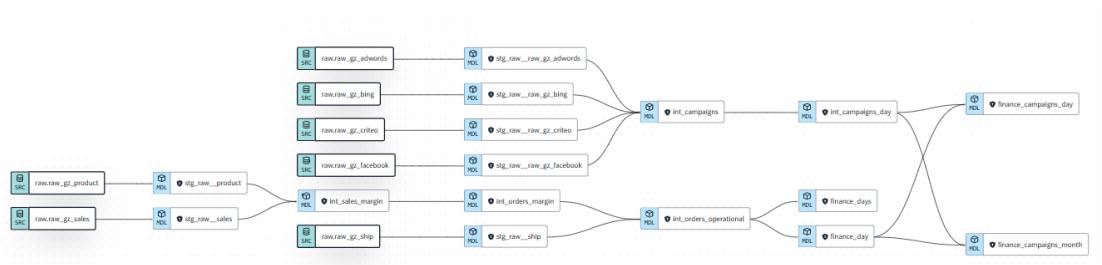
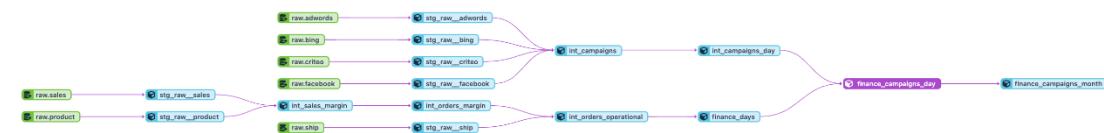
1. Write the query and build the model.

Your preview should look like this:

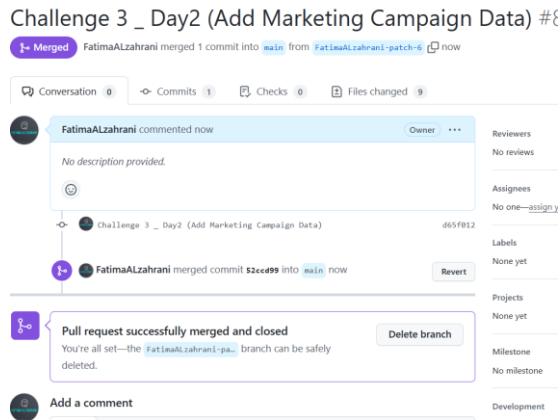
datemonth	ads_margin	average_basket	op
2021-09-01	258100.0	2292.4	39
2021-08-01	307729.0	2325.3	43
2021-07-01	323371.0	2304.1	44

The screenshot shows the DBT project interface with the file `finance_campaigns_month.sql` open. The code is a SQL SELECT statement that aggregates data from various sources into monthly metrics. Below the code is a preview window showing the results of the query, which matches the table provided above. The preview includes columns for `date_month`, `ads_margin`, `average_basket`, `operational_margin`, `ads_cost`, `impression`, `click`, and `quantity`. The results show four rows corresponding to the dates in the table, with values such as 258100.0 for ads\_margin and 2292.4 for average\_basket.

And your lineage like this :

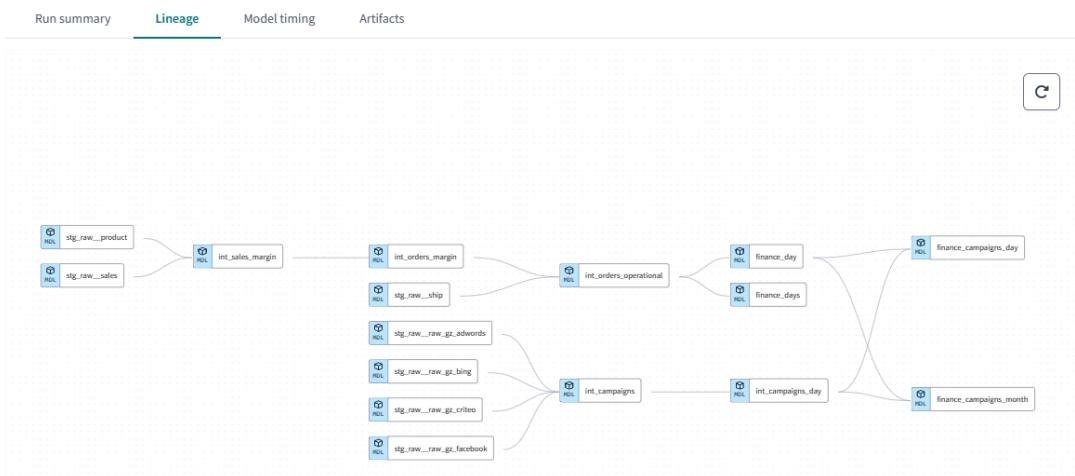


## 2. Merge to main with a pull request



## 3. Run the job in production environment

Look at your lineage in production



## Apply some Modification without Building the Complete Workflow.

Someone in charge of `facebook` raw data told you they changed the name of the `paid_source` and they ask you to update the mart data especially `finance_campaigns_day` before the next schedule update.

1. Define a new job corresponding to this task with a single command. Your job should be the lightest as possible in term of computation (to do not rerun models that don't change).

```
dbt build --select finance_day+
Ibt build --select finance_day+ success 9.2s
Ibt build --select finance_campaigns_day success 4.4s
Ibt build --select int_campaigns_day success 2.2s
Ibt build --select int_campaigns success 1.9s
Ibt build --select stg_raw__raw_gz_facebook success 1.8s
```

dbt build --select finance\_day+ success

Ibt FatimaALzahran-patch-6 1 minute ago

System Logs

All 3 Pass 3 Warn 0 Error 0 Skip 0 Running 0

finance\_day

finance\_campaigns\_day

finance\_campaigns\_month