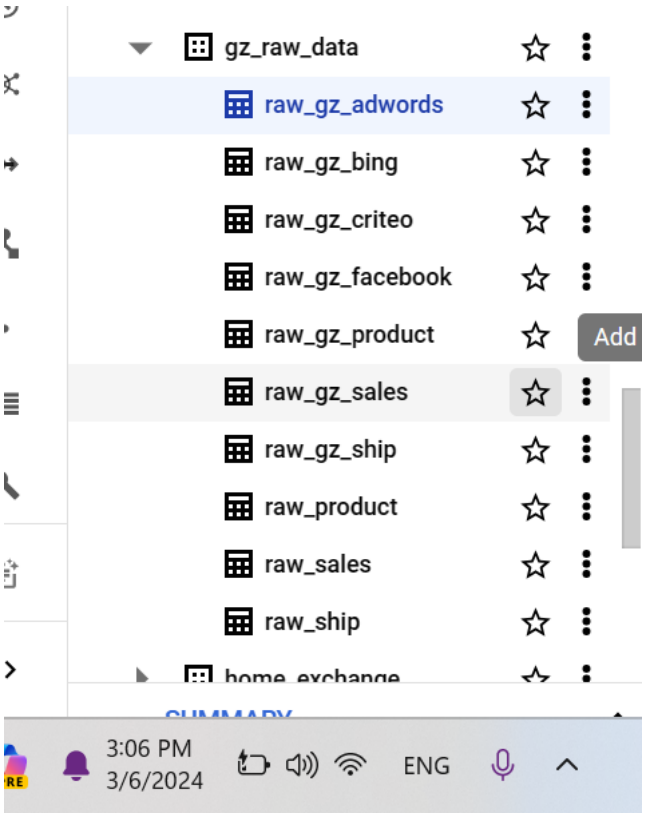
gz-dbt-repository/models at main · FatimaALzahrani/gz-dbt-repository (github.com)

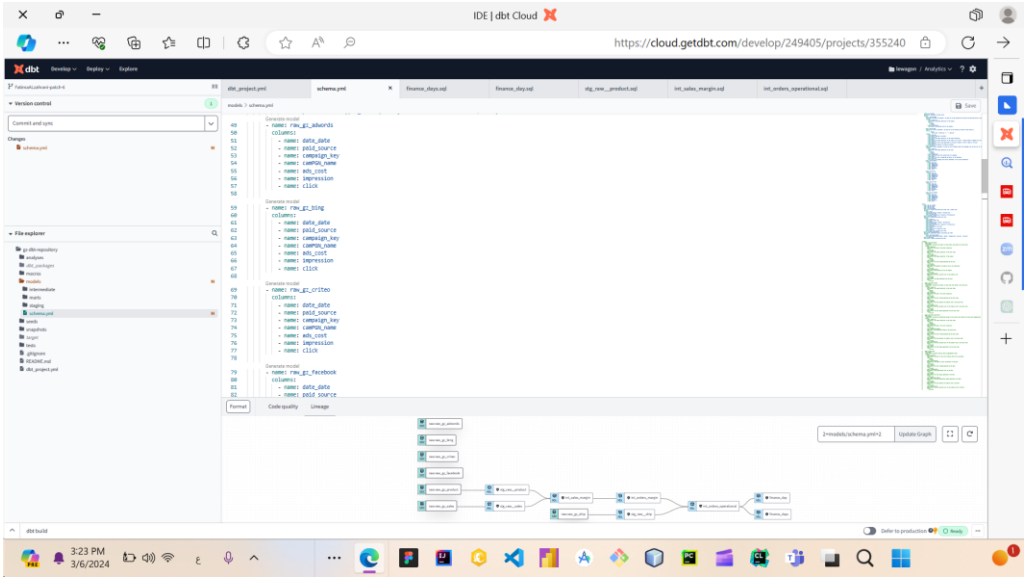
1 - Add the Source

1. Copy the following tables in your `gz_raw_data` dataset into your own BigQuery project:
   - raw_gz_adwords
   - raw_gz_bing
   - raw_gz_criteo
   - raw_gz_facebook

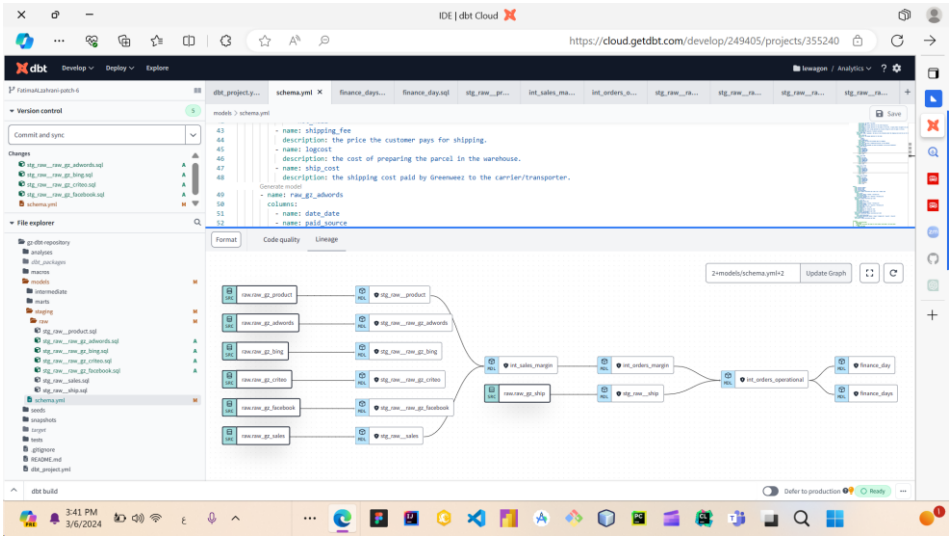
   Help - *Google documentation* to copy a table



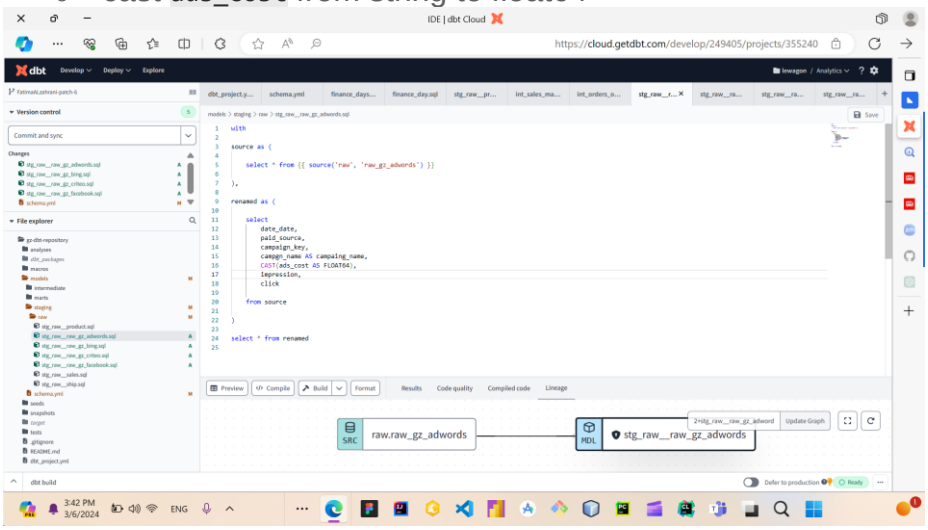
2. Add the source to your `schema.yml` in dbt cloud



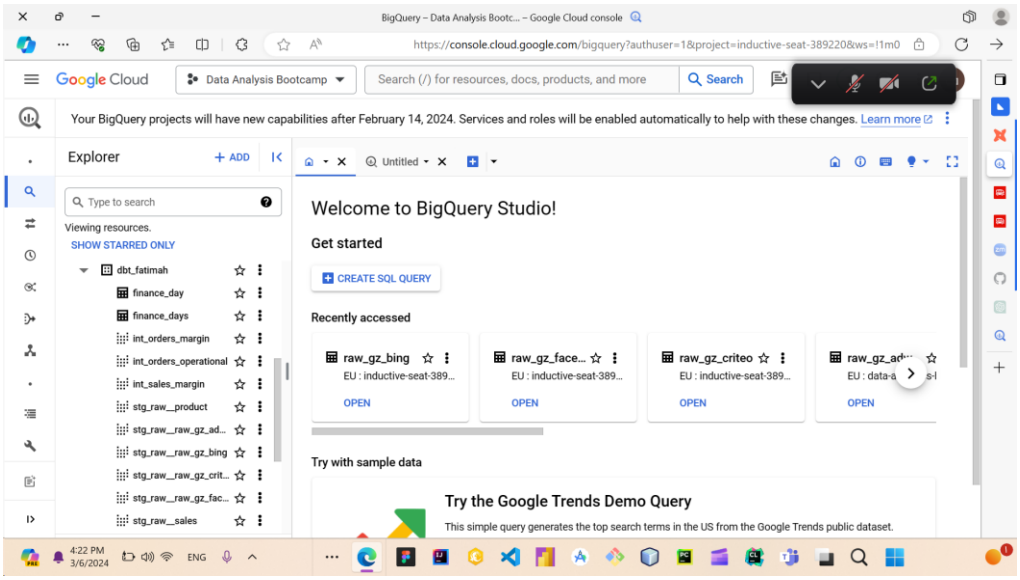
3. Generate staging models with the automatic process



4. Clean the staging models
    - rename `campaing_name` correctly
    - cast `ads_cost` from string to float64



5. Build all your staging sources



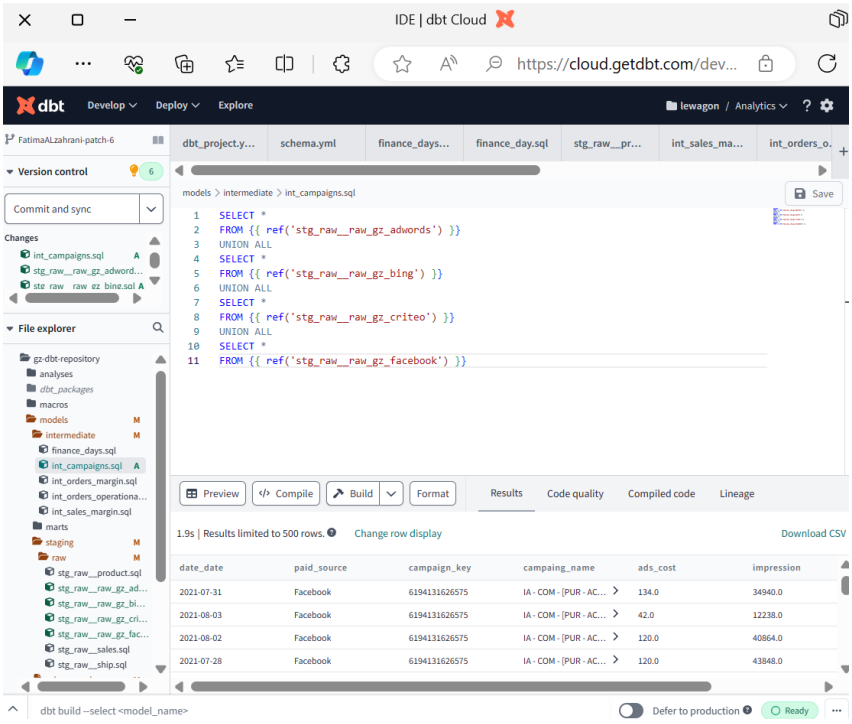
2 – Create `int_campaigns.sql` Model

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

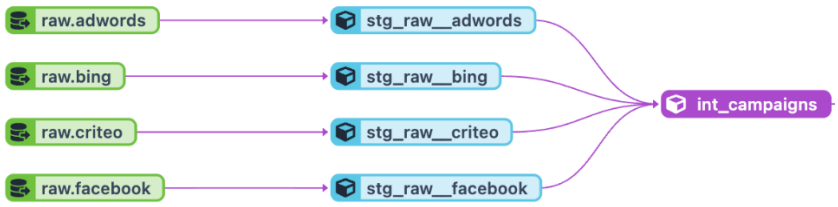
1. Make sure all the columns have the same name across the different tables, same for datatype.

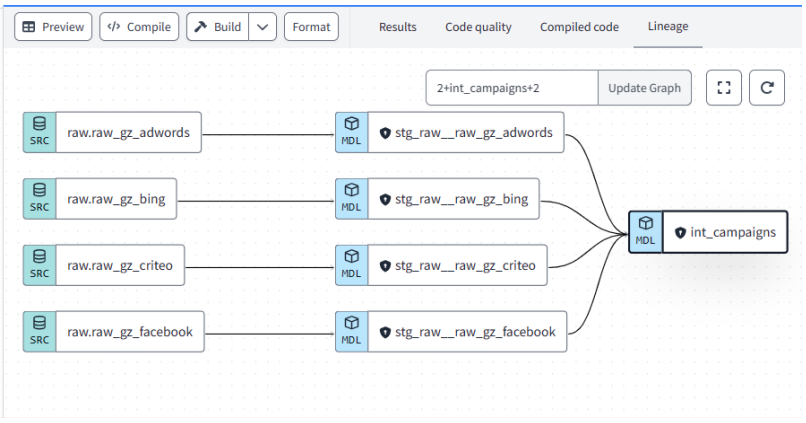
   Done

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



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`
    - Test primary key
    - Add relevant tests to other columns

models > schema.yml

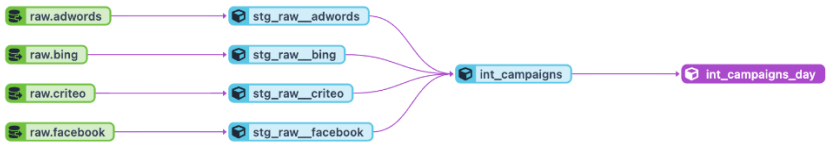
```yaml
- name: raw_gz_adwords
  description: raw_gz_adwords table
  tests:
      - unique:
          column_name: "(campaign_key || '-' || date_date)"
      - not_null
  columns:
    - name: date_date
      description: the date of purchase
    - name: paid_source
      description: The source from which the ad was paid for
    - name: campaign_key
      description: The unique identifier for the ad campaign.
    - name: camPGN_name
      description: The name or title of the ad campaign
    - name: ads_cost
      description: The cost incurred for running the ad campaign
    - name: impression
      description: The number of times the ad was displayed to users.
    - name: click
      description: The number of times users clicked on the ad
```

3 - Create `int_campaigns_day.sql` Model

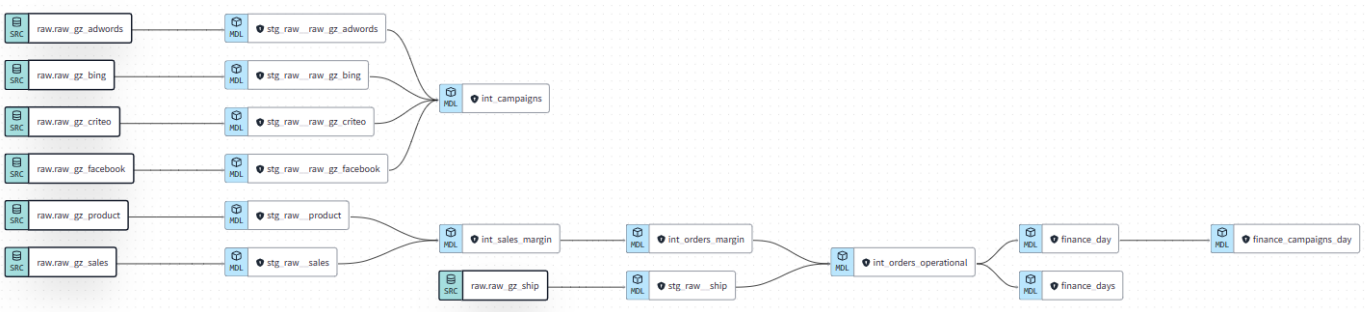
You want to create an aggregated version of `int_campaigns` model for each day.

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`



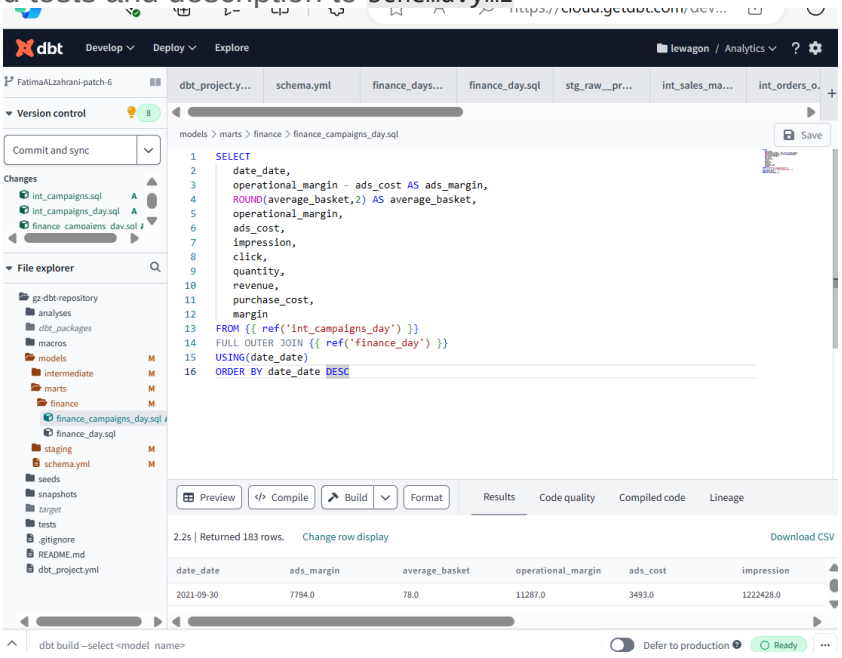
4 - Create `finance_campaigns_day.sql` Mart Model

1. 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.
   📌 Ads margin = operational margin - ads cost

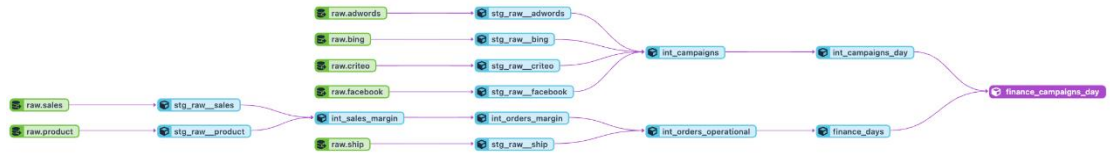
   Your preview should look like this:

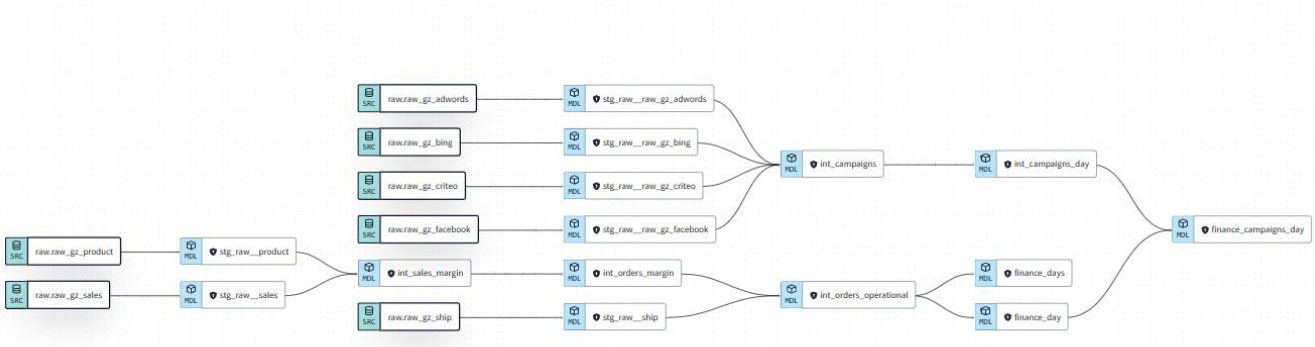
date_date	ads_margin	average_basket	operational_m...	a
2021-09-30	7794.0	78.4	11287.0	34
2021-09-29	6882.0	70.2	11708.0	48
2021-09-28	7925.0	73.7	13361.0	54

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



3. Check your lineage.





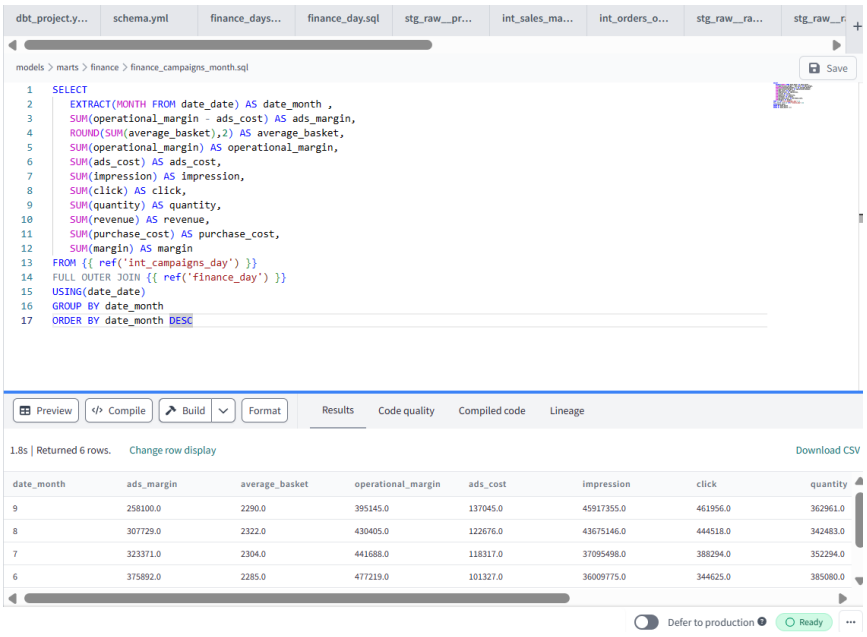
5 - Create `finance_campaigns_month.sql` Mart Model

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

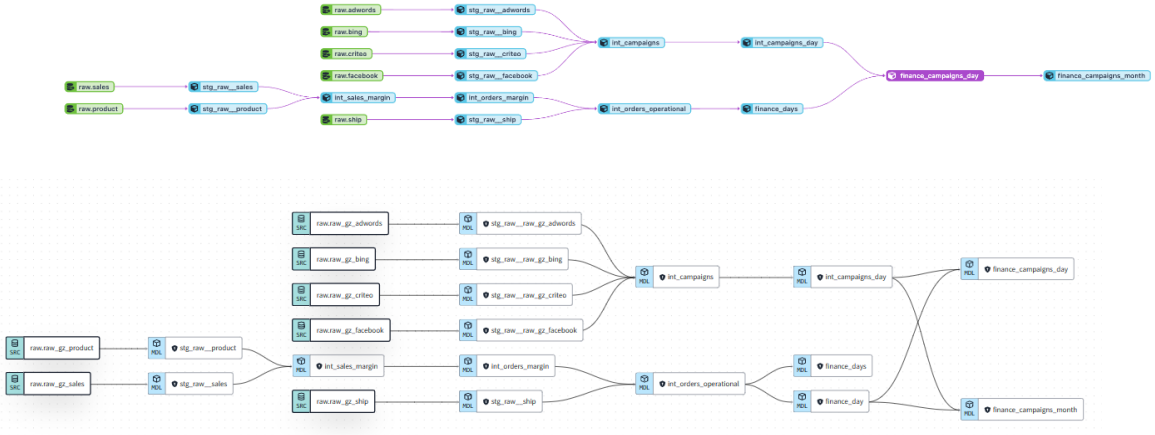
1. Write the query and build the model.

   Your preview should look like this:

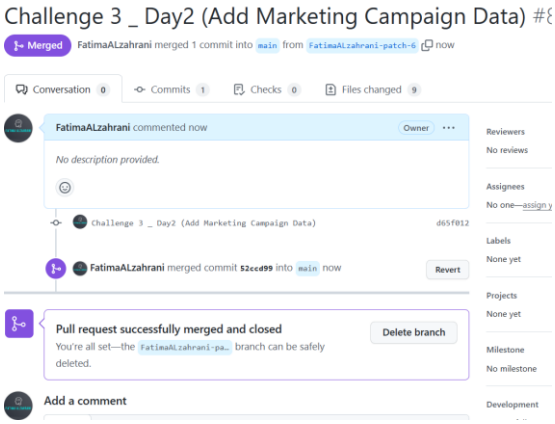
datemonth	ads_margin	average_basket	op
2021-09-01	258100.0	2292.4	39
2021-08-01	307729.0	2325.3	43
2021-07-01	323371.0	2304.1	44



   And your lineage like this :

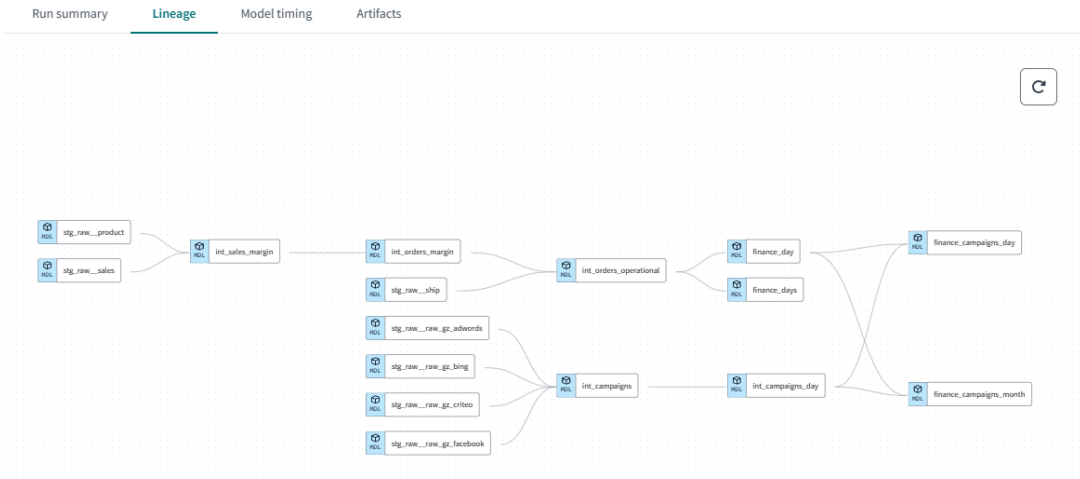


2. Merge to main with a pull request



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).