Google Ads
Batch Data Pipeline

By Shaheer Khan

Table of contents

01 Architecture

02 Problem Statement

03 Data Source: Google Ads

04 Data Ingestion: Airbyte

05 Data Transformation: dbt

06 Data Storage: Postgres

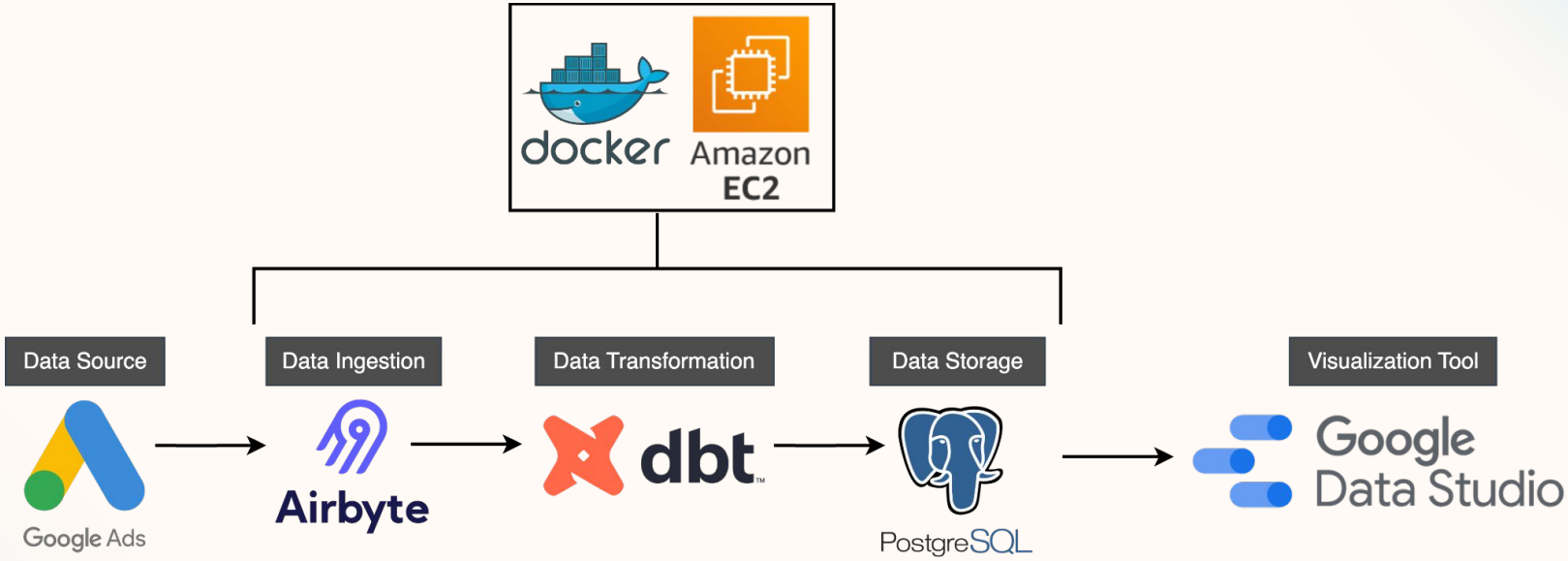
07 Visualization: Looker Studio

08 Future Iteration

01

Architecture

Architecture



Architecture: EC2 Ubuntu Instance

Tasks

Few key tasks and characteristics of EC2 Instance are discussed below:

1. Docker Compose was used to set up and run Postgres and Airbyte containers on EC2 Ubuntu Instance

2. To run the dbt models a shell script with crontab was used, a preview of the shell script is shown below alongside the crontab script. The dbt models are scheduled to run at 6.30 PM EST. In the next slide we will discuss the output of the crontab script

run_dbt.sh

```
#!/bin/bash
source /home/ubuntu/dbt/beamdata_dbt/.env
cd /home/ubuntu/dbt
source /home/ubuntu/dbt/dbtenv/bin/activate
cd /home/ubuntu/dbt/beamdata_dbt
dbt run
```

```
CRON_TZ="America/New_York"
30 18 * * * echo "Starting dbt run at $(date +'\%Y-\%m-\%d \%H:\%M:\%S')" >>
/home/ubuntu/crondbtlogfile.log && /home/ubuntu/run_dbt.sh >> /home/ubuntu/crondbtlogfile.log 2>&1 &&
echo "Finished dbt run at $(date +'\%Y-\%m-\%d \%H:\%M:\%S')" >> /home/ubuntu/crondbtlogfile.log
```

```
GNU nano 6.2                                                    /tmp/crontab.zaMQ35/crontab
@RON_TZ="America/New_York"
30 18 * * * echo "Starting dbt run at $(date +'\%Y-\%m-\%d \%H:\%M:\%S')" >> /home/ubuntu/crondbtlogfile.log && /home/ubuntu/run_dbt.sh >> /home/ubuntu/crondbtlogfile.log 2>&1 && echo "Finished dbt run at $(date +'\%Y-\%m-\%d \%H:\%M:\%S')" >> /home/ubuntu/crondbtlogfile.log
```

Architecture: EC2 Ubuntu Instance

Tasks

To discuss more about crontab and output of log files exported:

2. The output and progress of the dbt models that were executed using a shell script with crontab was stored in a log file with start date and time, the models run progress and finish date and time as shown on the right

crondbtlogfile.log

```
[0m23:34:40  Done. PASS=15 WARN=0 ERROR=0 SKIP=0 TOTAL=15
Finished dbt run at 2024-11-28 18:34:41
Starting dbt run at 2024-11-29 18:30:01
[0m23:30:04  Running with dbt=1.7.9
[0m23:30:04  Registered adapter: postgres=1.7.9
[0m23:30:04  Found 17 models, 127 tests, 6 sources, 0 exposures, 0 metrics, 410 macros, 0 groups, 0 semantic models
[0m23:30:04
[0m23:30:05  Concurrency: 1 threads (target='dev')
[0m23:30:05
[0m23:30:05  1 of 17 START sql view model gads_dev.stg_gads__ad_group ....................... [RUN]
[0m23:30:05  1 of 17 OK created sql view model gads_dev.stg_gads__ad_group .................. [[32mCREATE VIEW[0m in 0.21s]
[0m23:30:05  2 of 17 START sql view model gads_dev.stg_gads__ad_group_ad ................... [RUN]
[0m23:30:05  2 of 17 OK created sql view model gads_dev.stg_gads__ad_group_ad ............... [[32mCREATE VIEW[0m in 0.08s]
[0m23:30:05  3 of 17 START sql table model gads_dev.stg_gads__ad_group_ad_legacy ........... [RUN]
[0m23:30:05  3 of 17 OK created sql table model gads_dev.stg_gads__ad_group_ad_legacy ...... [[32mSELECT 29903[0m in 0.29s]
[0m23:30:05  4 of 17 START sql view model gads_dev.stg_gads__campaign ....................... [RUN]
[0m23:30:05  4 of 17 OK created sql view model gads_dev.stg_gads__campaign ................. [[32mCREATE VIEW[0m in 0.07s]
[0m23:30:05  5 of 17 START sql view model gads_dev.stg_keywords__ad_group ................... [RUN]
[0m23:30:05  5 of 17 OK created sql view model gads_dev.stg_keywords__ad_group ............. [[32mCREATE VIEW[0m in 0.07s]
[0m23:30:05  6 of 17 START sql view model gads_dev.stg_keywords__ad_group_criterion ......... [RUN]
[0m23:30:05  6 of 17 OK created sql view model gads_dev.stg_keywords__ad_group_criterion .... [[32mCREATE VIEW[0m in 0.06s]
[0m23:30:05  7 of 17 START sql view model gads_dev.stg_keywords__campaign ................... [RUN]
[0m23:30:05  7 of 17 OK created sql view model gads_dev.stg_keywords__campaign ............. [[32mCREATE VIEW[0m in 0.07s]
[0m23:30:05  8 of 17 START sql view model gads_dev.stg_keywords__keyword ..................... [RUN]
[0m23:30:06  8 of 17 OK created sql view model gads_dev.stg_keywords__keyword ............... [[32mCREATE VIEW[0m in 0.07s]
[0m23:30:06  9 of 17 START sql view model gads_dev.int_ad_names_extracted ................... [RUN]
[0m23:30:06  9 of 17 OK created sql view model gads_dev.int_ad_names_extracted ............. [[32mCREATE VIEW[0m in 0.06s]
[0m23:30:06  10 of 17 START sql incremental model gads_dev.incr_gads__ad_group_ad_legacy .... [RUN]
[0m23:30:06  10 of 17 OK created sql incremental model gads_dev.incr_gads__ad_group_ad_legacy  [[32mINSERT 0 1782[0m in 0.22s]
[0m23:30:06  11 of 17 START sql view model gads_dev.int_wide_table_base_summary_table ....... [RUN]
[0m23:30:06  11 of 17 OK created sql view model gads_dev.int_wide_table_base_summary_table .. [[32mCREATE VIEW[0m in 0.06s]
[0m23:30:06  12 of 17 START sql table model gads_dev.keyword_x_campaign_x_ad_group_step1 .... [RUN]
[0m23:34:09  12 of 17 OK created sql table model gads_dev.keyword_x_campaign_x_ad_group_step1  [[32mSELECT 1026743[0m in 242.93s]
[0m23:34:09  13 of 17 START sql view model gads_dev.int_wide_table_base_summary_table_w_increm  [RUN]
[0m23:34:09  13 of 17 OK created sql view model gads_dev.int_wide_table_base_summary_table_w_increm  [[32mCREATE VIEW[0m in 0.07s]
[0m23:34:09  14 of 17 START sql table model gads_dev.campaign_x_ad_group_x_ad_groud_ad_bigtable  [RUN]
[0m23:34:24  14 of 17 OK created sql table model gads_dev.campaign_x_ad_group_x_ad_groud_ad_bigtable  [[32mSELECT 20536[0m in 15.06s]
[0m23:34:24  15 of 17 START sql table model gads_dev.keyword_x_campaign_x_ad_group_step2 .... [RUN]
[0m23:34:31  15 of 17 OK created sql table model gads_dev.keyword_x_campaign_x_ad_group_step2  [[32mSELECT 948726[0m in 7.12s]
[0m23:34:31  16 of 17 START sql table model gads_dev.campaign_x_ad_group_x_ad_groud_ad_bigtable_w_increm  [RUN]
[0m23:34:46  16 of 17 OK created sql table model gads_dev.campaign_x_ad_group_x_ad_groud_ad_bigtable_w_increm  [[32mSELECT 20536[0m in 15.06s]
[0m23:34:46  17 of 17 START sql table model gads_dev.keyword_x_campaign_x_ad_group_UI ....... [RUN]
[0m23:34:59  17 of 17 OK created sql table model gads_dev.keyword_x_campaign_x_ad_group_UI .. [[32mSELECT 1062854[0m in 12.91s]
[0m23:34:59
[0m23:34:59  Finished running 10 view models, 6 table models, 1 incremental model in 0 hours 4 minutes and 54.64 seconds (294.64s).
[0m23:34:59
[0m23:34:59  [32mCompleted successfully[0m
[0m23:34:59
[0m23:34:59  Done. PASS=17 WARN=0 ERROR=0 SKIP=0 TOTAL=17
Finished dbt run at 2024-11-29 18:34:59
```

02

Problem Statement

Problem Statement

Tasks

The project encompasses main aspects of a batch data pipeline, with the goal of reducing manual data pulls from Google Ads report builder and automating the data views to be exported to Looker Studio on a set cadence for Campaign and Keyword analysis.

The data source used in the project is Google Ads api via Airbyte

Part A: Data Infrastructure

Utilize Docker Compose to deploy and manage Airbyte, dbt, and Postgres containers on an AWS EC2 instance

Problem Statement

Part B: Data Extraction and Processing

1. Data is extracted using Airbyte from Google Ads API (Airbyte version 3.7.1)
2. dbt is used to create staging, intermediate, and mart models for transformation of raw data
3. Two Big Tables is the final output of the dbt mart models to depict the Ad and Keyword views found in the UI of the Google Ads Reporting

Part C: Data Storage, Analysis & Visualization

The output of dbt models are stored in Postgres where dbt models are stored as views or tables. The Two Big Tables are exported to Looker Studio for analysis and reporting

03

Data Source: Google Ads

Data Source: Google Ads

Tasks

The views that were expected to be the final output of the data pipeline are as follows:

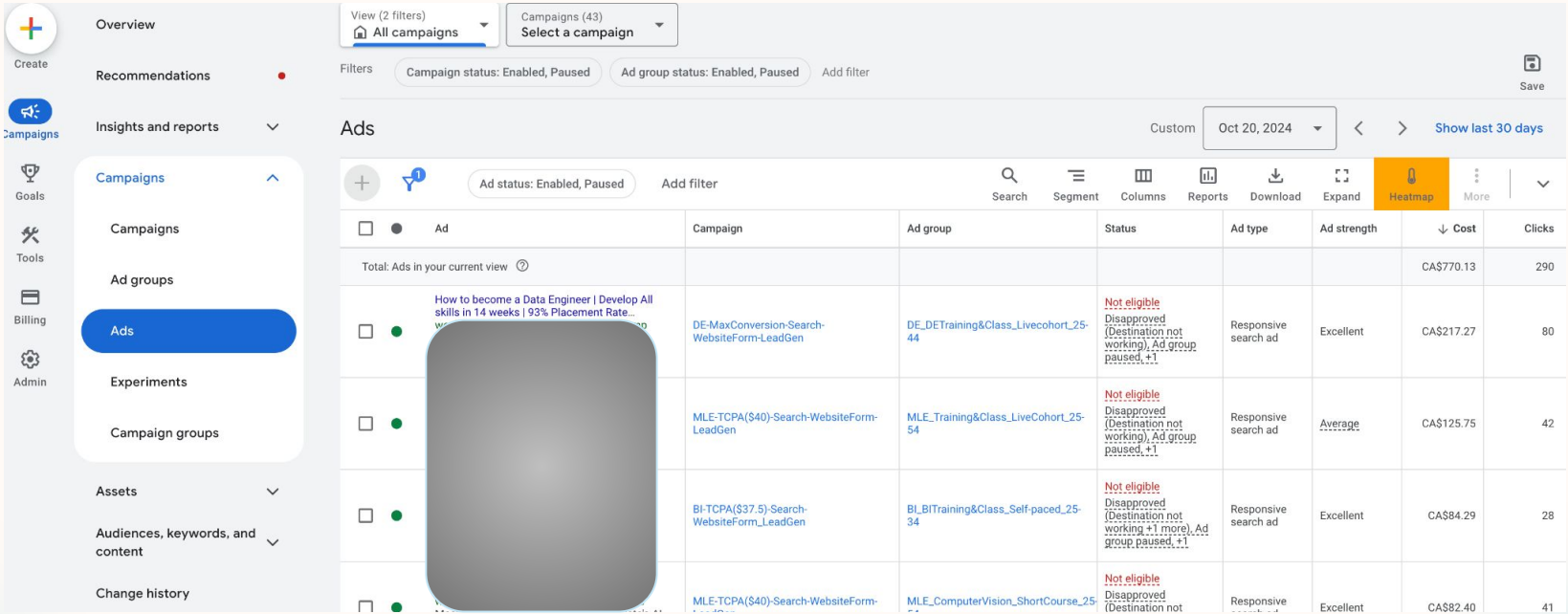
1. **Ad View:** This view is mainly comprised of these dimensions; Ad Name, Campaign Name, Ad Group and Status alongside key metrics such as Cost, Clicks etc. This is also shown in the next slide (Slide 12)
2. **Keyword View:** This view is mainly comprised of these dimensions; Keyword, Match Type, Campaign Name, Ad Group and Status alongside key metrics such as Cost, Clicks, Impressions, Interactions etc. This is also shown on Slide 13

Approach/Next Steps

The next step is to evaluate which tables from the API would be required to create the 2 views mentioned above. This will be discussed in the next section (Section 4: Data Ingestion)

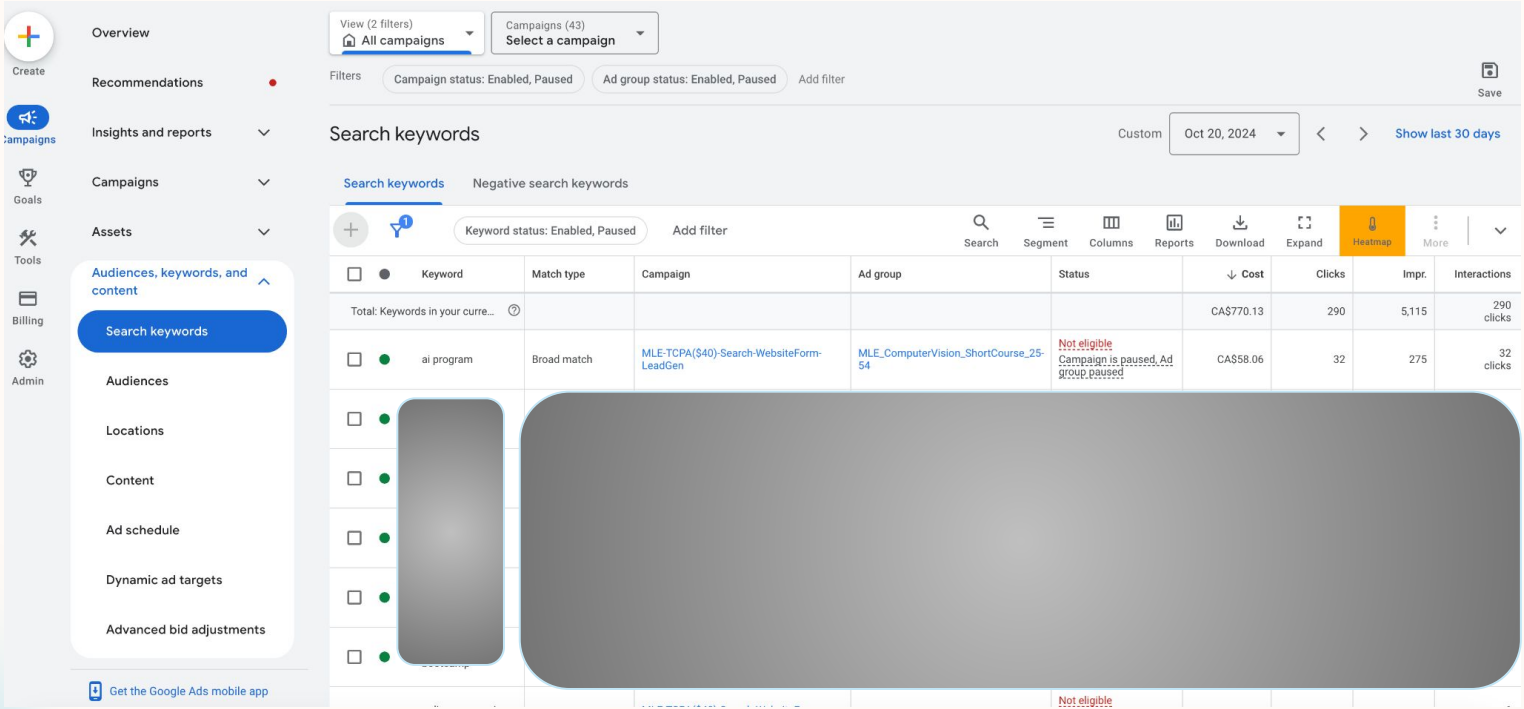
Data Source: Google Ads

Ad View



Data Source: Google Ads

Keyword View



04

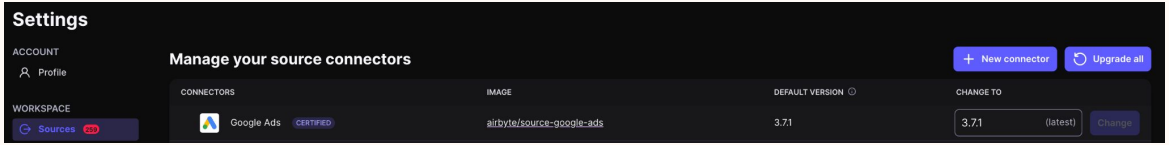
Data Ingestion: Airbyte

Data Ingestion: Airbyte

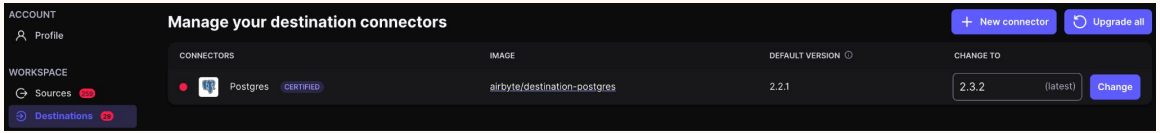
Tasks

The following tasks were carried out to setup connectors for source and destination in Airbyte:

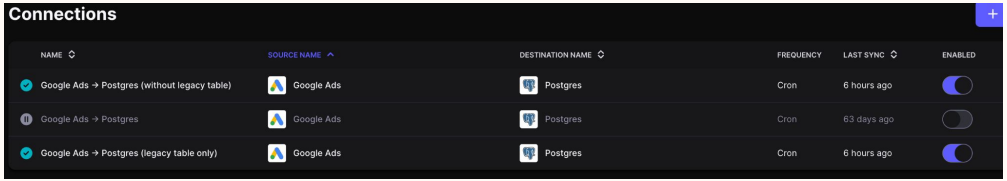
1. Source Connector was setup for Google Ads



2. Destination Connector was setup to export raw data from the Google Ads connector to Postgres destination



3. Connections were set up in Airbyte as shown below. Note: Due to an error in one of the tables, 2 connections were created to isolate the table causing import error. More details can be found in the coming slides.



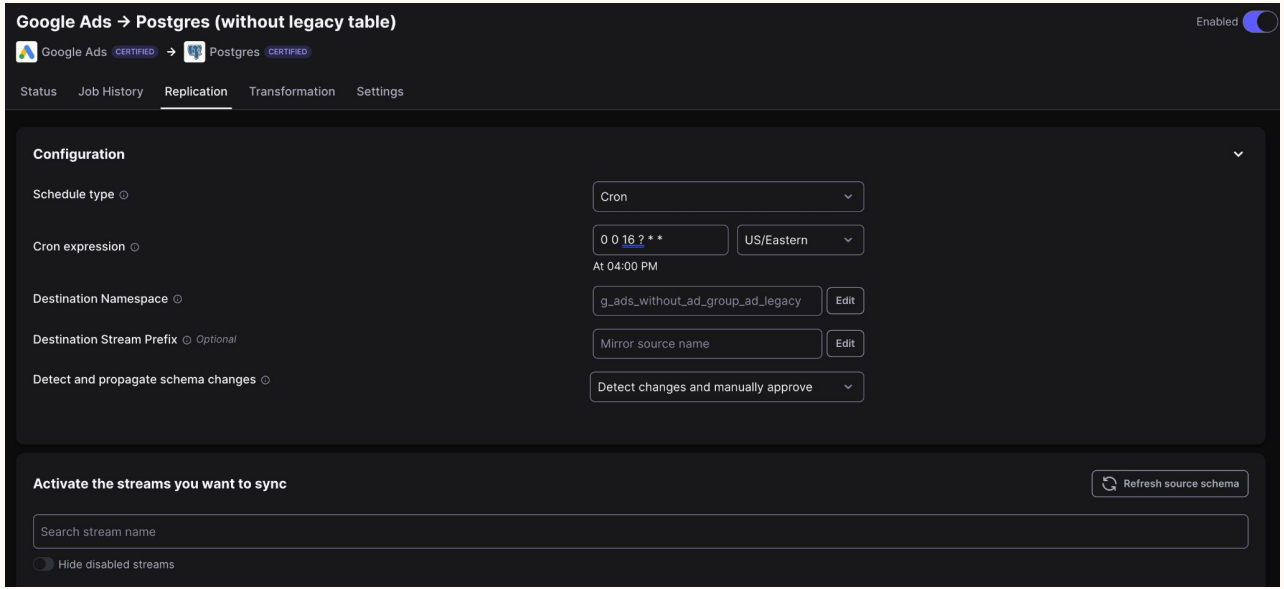
Data Ingestion: Airbyte

4. Both connections were set up with a cron job running daily at 4pm EST and with the following configurations as shown below.

   Only 1 of 2 connections is shown below

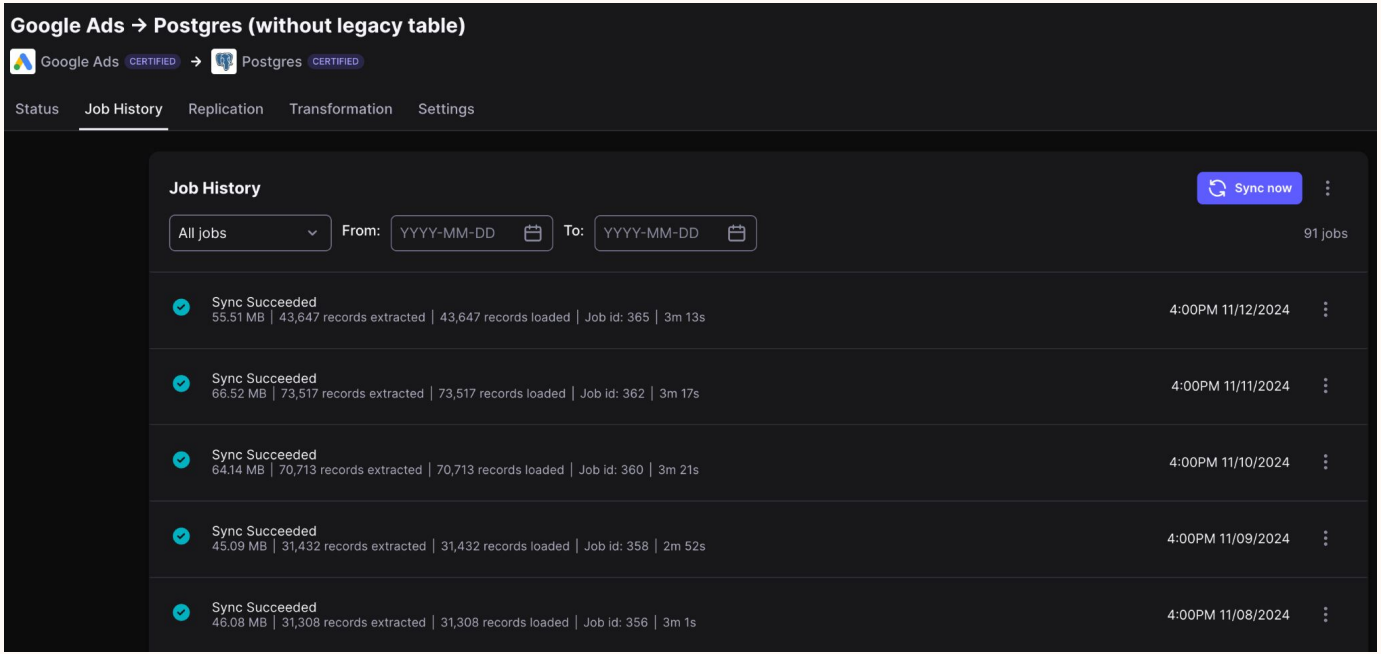
Note:

- If there was a schema change in the data source then manual approval would be done to maintain uptime of the pipeline



Data Ingestion: Airbyte

5. A sample output of raw data extraction and data ingestion in Postgres as scheduled to run daily is shown below

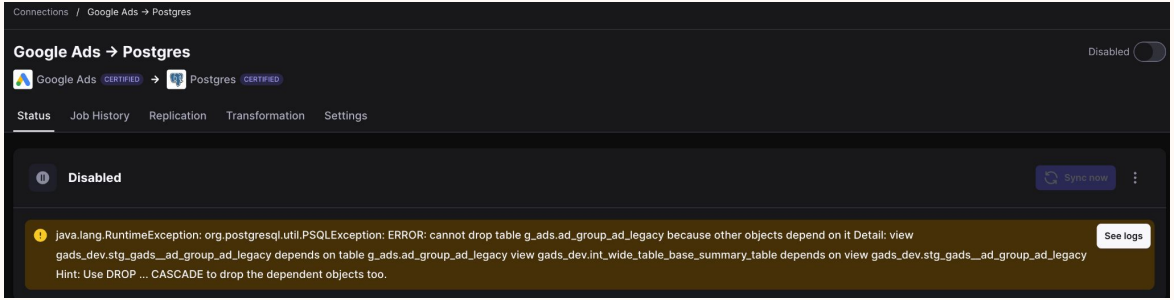


Data Ingestion: Airbyte

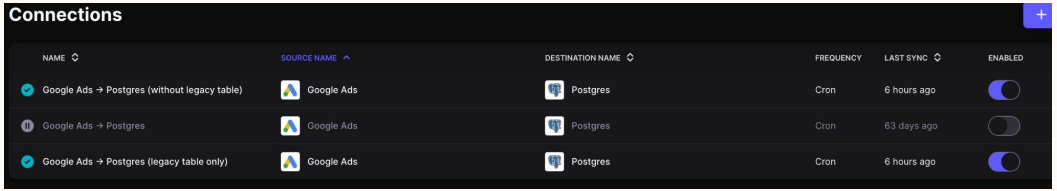
Issues Encountered

Upon dbt models running, there was an issue with one of the tables (streams known in Airbyte) called "ad_group_ad_legacy" that was causing the pipeline to fail. More detail can be found below:

1. Here is the error encountered in Airbyte after running dbt models and re-syncing Google Ads data



2. To solve this, 2 separate connections were made in Airbyte one with the table (stream) called "ad_group_ad_legacy" and the other with all tables except for "ad_group_ad_legacy" as shown below



05

Data Transformation: dbt

Data Transformation: dbt

Tasks

The following tasks were carried out to setup and run dbt models on Google Ads data:

1. For the ELT process a multi-layered data architecture is used, called "staging-intermediate-mart"

   Staging Layer: Raw, minimally transformed data from source systems.
   Intermediate Layer: Heavily transformed data, preparing it for analytical use.
   Mart Layer: Final, business-facing data, ready for analysis and reporting.

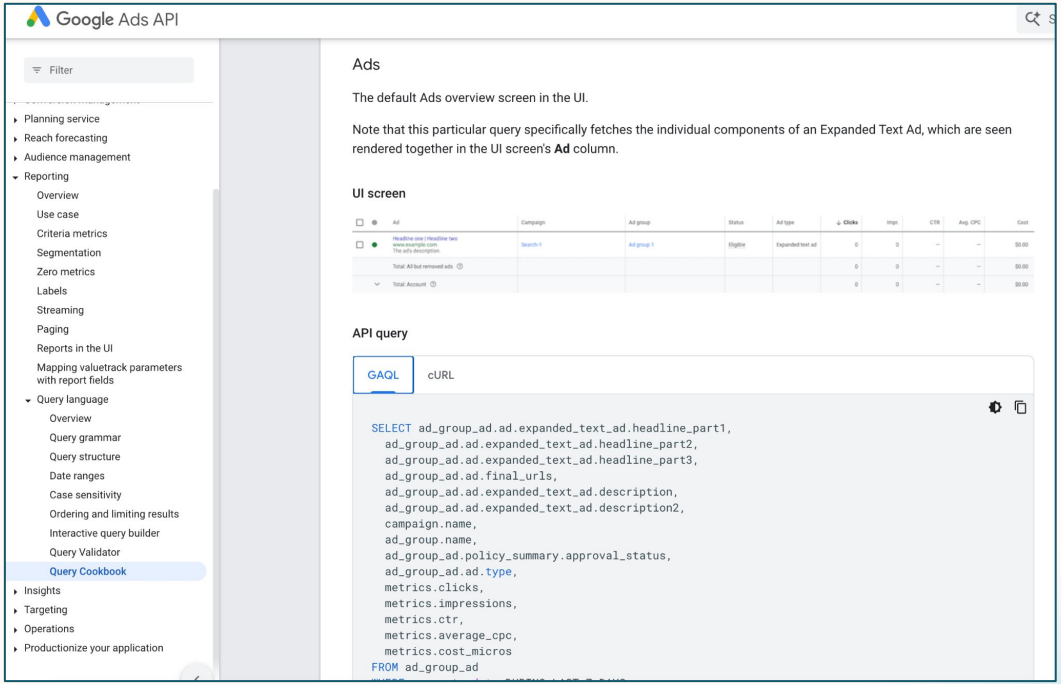
2. The Mart Layer will have the final tables depicting the **Keyword** and **Ad View** as shown in slides 12 and 13

Data Transformation: dbt

Tasks

3. To produce the Mart Layer, correct scoping of tables to be ingested for Staging Layer was carried out. Google Ads API documentation provided the tables, dimension, and metrics required to depict the UI view as mentioned in previous steps.

   The image shown on the right depicts Ad View in Google Ads, Keyword view can also be extracted from the same resource linked here



Data Transformation: dbt

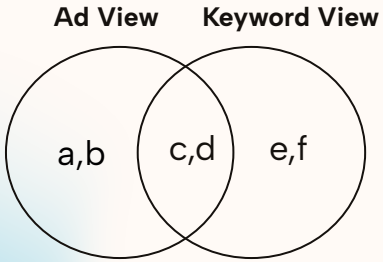
Now the structure of the dbt project will be discussed:

4. Staging Layer

   Raw, minimally transformed data from source systems.

The following tables were used:

a. ad_group_ad_legacy
b. ad_group_ad
c. ad_group
d. campaign
e. keyword_view
f. ad_group_criterion



5. Intermediate Layer

   Heavily transformed data, preparing it for analytical use.

Here, **major transformations** were executed for **Ad View** tables as the data required for reporting was not in a suitable format.

For Ad View, ad names were extracted for the Ad, from ad_group_ad table. The dimension that has this information is called ad_group_ad_ad_responsive_search_ad_headlines. Now the ad names are found in a list format in ad_group_ad_ad_responsive_search_ad_headlines dimension shown below

ad_group_ad_ad_id bigint	ad_group_ad_ad_responsive_search_ad_headlines jsonb
[redacted]	{"text: \"DevOps Engineer Bootcamp\"\nasset_performance_label: LEARNING\npolicy_summary_info {\n review_status: REVIEWED\n approval_status: APPROVED\n}\n", "text: \"DevOps Training\"\nas
	{"text: \"DevOps Engineer Bootcamp\"\nasset_performance_label: LEARNING\npolicy_summary_info {\n review_status: REVIEWED\n approval_status: APPROVED\n}\n", "text: \"DevOps Training\"\nas
	{"text: \"DevOps Engineer Bootcamp\"\nasset_performance_label: LEARNING\npolicy_summary_info {\n review_status: REVIEWED\n approval_status: APPROVED\n}\n", "text: \"DevOps Training\"\nas
	{"text: \"DevOps Engineer Bootcamp\"\nasset_performance_label: LEARNING\npolicy_summary_info {\n review_status: REVIEWED\n approval_status: APPROVED\n}\n", "text: \"DevOps Training\"\nas
	{"text: \"DevOps Engineer Bootcamp\"\nasset_performance_label: LEARNING\npolicy_summary_info {\n review_status: REVIEWED\n approval_status: APPROVED\n}\n", "text: \"DevOps Training\"\nas

In the next slide, the details regarding transformations required to extract ad names from ad_group_ad_ad_responsive_search_ad_headlines dimension for each Ad will be discussed

Data Transformation: dbt

To discuss more about the the transformations in the **Intermediate Layer**

5. Intermediate Layer

As shown in the previous slide, to extract ad names, from ad_group_ad table for **Ad View** we need to extract the ad names found in a list format in ad_group_ad_ad_responsive_search_ad_headlines dimension.

As shown in the transformed and newly created dimension called **"aggregated_texts"** below which stores an array of ad name text values extracted from a list of json values found in ad_group_ad_ad_responsive_search_ad_headlines source dimension.

As a final step of transformations, ad names for each Ad was extracted in separate dimensions from the "aggregated_texts" dimension as shown below, the new dimensions are called "ad_group_ad_ad_name_1", "ad_group_ad_ad_name_2" and more as needed.

	ad_group_ad_ad_id bigint	ad_group_ad_ad_responsive_search_ad_headlines jsonb	aggregated_texts text[]	ad_group_ad_ad_name_1 text	ad_group_ad_ad_name_2 text
1		["text: \"DevOps Engineer Bootcamp\"\nasset_perf...	{"DevOps Engineer Bootcamp","DevOps Training","Become a DevOps Engineer...	DevOps Engineer Bootcamp	DevOps Training
2		["text: \"Data Science BootCamp 2024\"\nasset_pe...	{"Data Science BootCamp 2024","14 Week Data Science Learning","Best Data ...	Data Science BootCamp 2024	14 Week Data Science Learn...
3		["text: \"Become A Data Analyst\"\nasset_perform...	{"Become A Data Analyst","Data Analytics Bootcamp","Data Analyst Part Time...	Become A Data Analyst	Data Analytics Bootcamp
4		["text: \"Machine Learning Bootcamp\"\nasset_per...	{"Machine Learning Bootcamp","Machine Learning Courses","Machine Learni...	Machine Learning Bootcamp	Machine Learning Courses
5		["text: \"How to become a Data Engineer\"\nasset_...	{"How to become a Data Engineer","Develop All skills in 14 weeks","93% Place...	How to become a Data Engineer	Develop All skills in 14 weeks

Data Transformation: dbt

For the last stage of the ELT process in "staging-intermediate-mart" architecture, we will discuss **Mart Layer**

6. Mart Layer

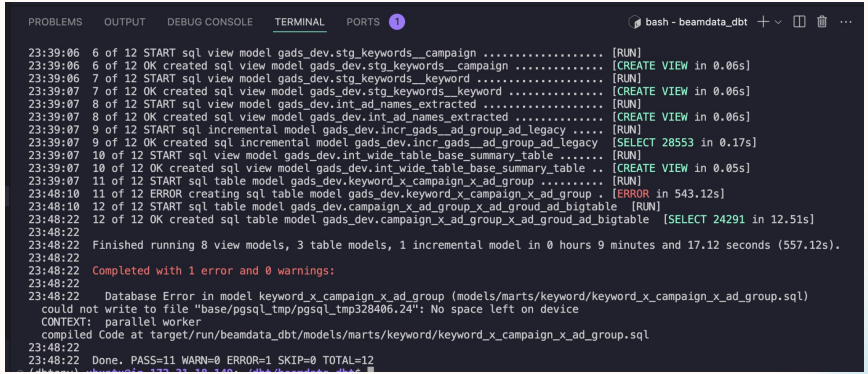
   This is the final, business-facing data, ready for analysis and reporting layer.

In this layer there are 3 sublayers:

I. **Ad View:** This has the Big Table depicting the Ad View in Google Ads website
II. **Keyword View:** This has the Big Table depicting the Keyword View in Google Ads website
III. **Incremental Model:** This is an incremental model for ad_group_ad_legacy table from source

For **Keyword View Big Table** due to the size of the model and the size of the EC2 instance the model had to be broken down in 3 models, without which we were running into an error as shown on the right.

More details on this fix is shown in the next slide

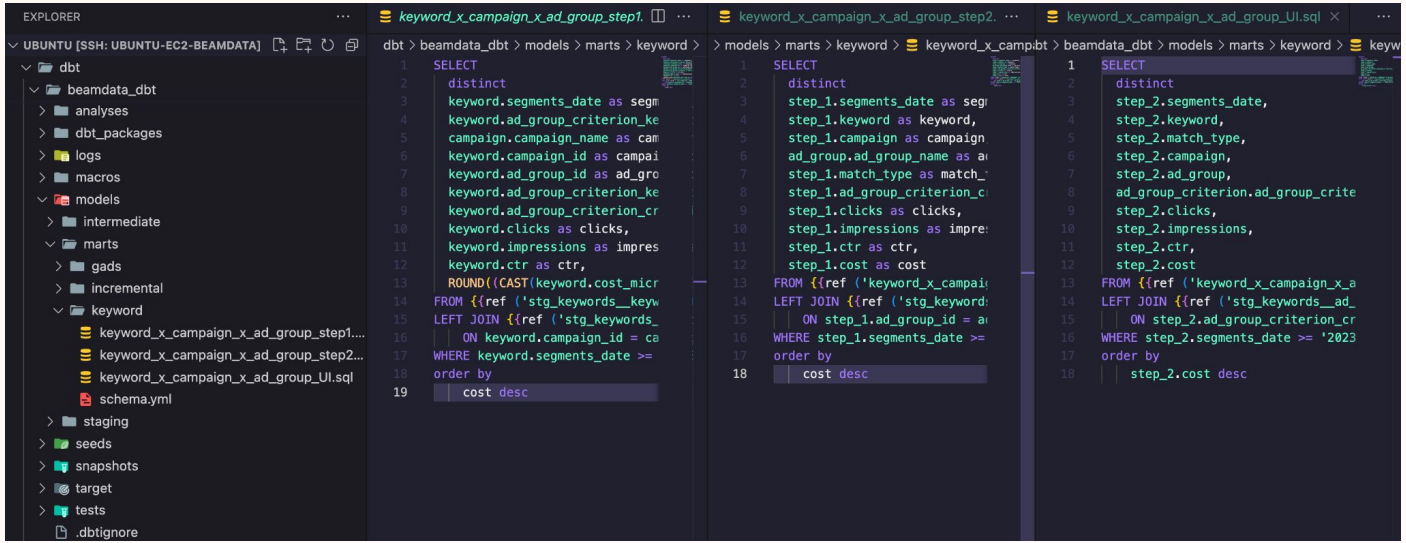


Data Transformation: dbt

To discuss more about the debugging of error in the sublayer of **Keyword Mart Layer**

6. Mart Layer –> Keyword Mart Layer

A solution to the large processing required for **Keyword Mart Layer** was to break the model in 3 subsequent models as shown below. The final output model which is the Big Table to show Keyword View is named "keyword_x_campaign_x_ad_group_UI" as shown below



Data Transformation: dbt

Lastly, to discuss about the last sublayer in **Mart Layer** called **Incremental Model**

6. Mart Layer –> Incremental Model

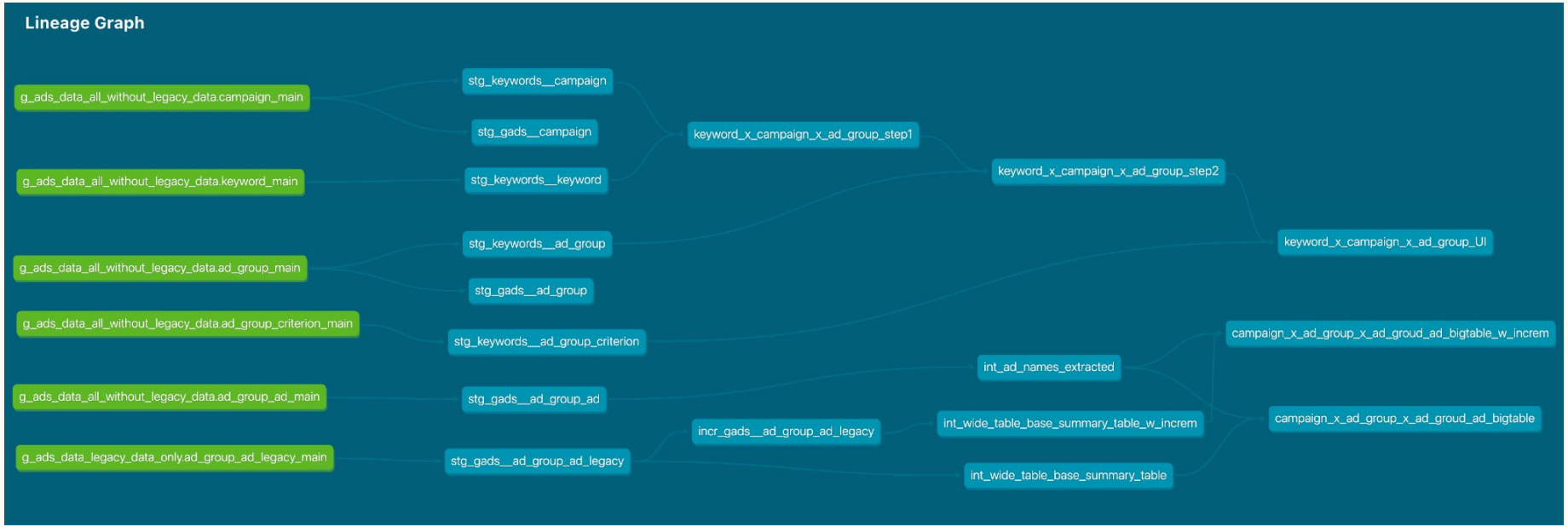
For Performance Optimizations and Cost Efficiency, as full rebuilds are expensive and data warehouses charge based on compute usage hence incremental processing is chosen so that only the new or updated records are processed

Few key configurations used with justification are as follows:

- Incremental strategy called 'delete+insert' is used instead of 'merge' due to a lack of unique key as the nature of the data set didn't allow for a creation of primary or surrogate key after using a combination of dimension to implement unique key hence the 'merge' strategy was not used
- Incremental Model has a look-back window of 14 days, so that the data is added for the last 14 days
- Batch IDs were incorporated to identify each incremental run and to help with debugging incremental run outputs

	batch_id text	segments_date date	campaign_name character varying
60	0305ef6a-64af-4de3-8c49-276c06788c12	2024-10-22	MLE-TCPA($40)-Search-WebsiteForm-LeadGen
61	0305ef6a-64af-4de3-8c49-276c06788c12	2024-11-03	DE-MaxConversion-Search-WebsiteForm-LeadG…
62	0305ef6a-64af-4de3-8c49-276c06788c12	2024-10-28	DS-TCPA($37.5)-Search-WebsiteForm-LeadGen
63	0305ef6a-64af-4de3-8c49-276c06788c12	2024-10-28	BI-TCPA($37.5)-Search-WebsiteForm_LeadGen
64	0305ef6a-64af-4de3-8c49-276c06788c12	2024-11-02	MLE-TCPA($40)-Search-WebsiteForm-LeadGen
65	1bb6fb76-e8c9-497a-a076-bed447f70e29	2024-11-08	Fundamental_search_webform_LenGen
66	1bb6fb76-e8c9-497a-a076-bed447f70e29	2024-11-06	Branding_Video_Traffic
67	1bb6fb76-e8c9-497a-a076-bed447f70e29	2024-11-02	DE-MaxConversion-Search-WebsiteForm-LeadG…
68	1bb6fb76-e8c9-497a-a076-bed447f70e29	2024-11-05	AI&MLELearningTrack_Search_WebForm_LeadG…

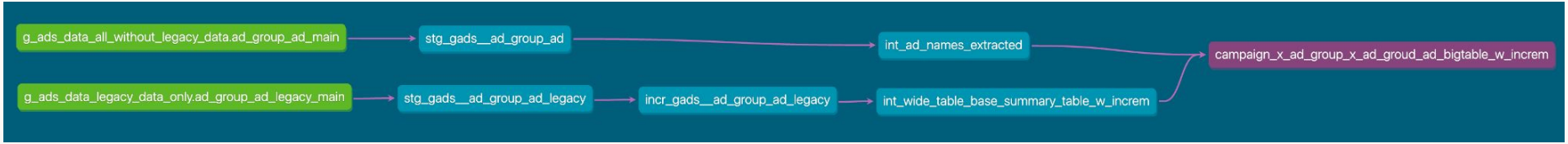
Data Transformation: dbt



Data Transformation: dbt

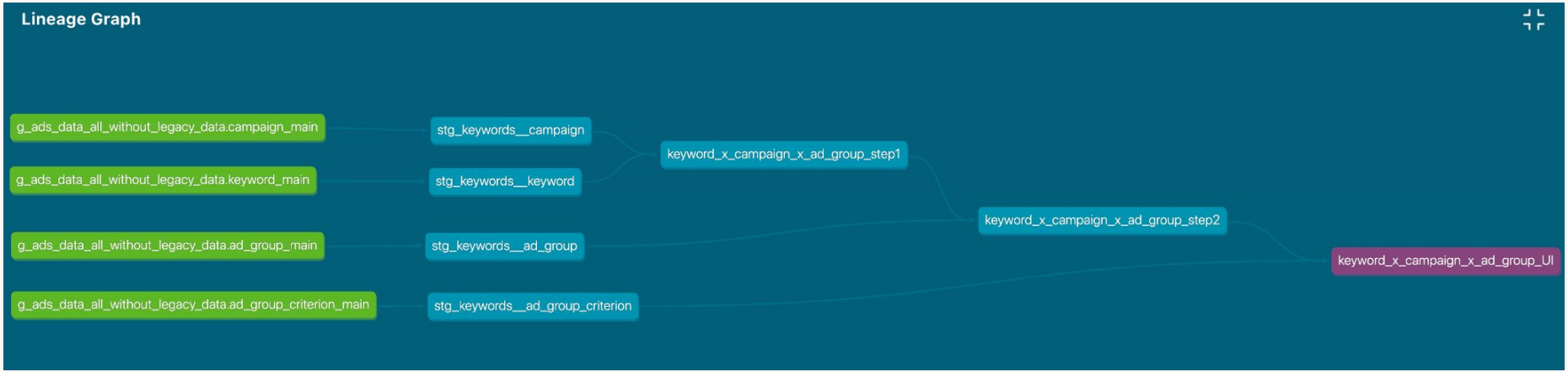
Ad View

Ad View with Incremental Model



Data Transformation: dbt

Keyword View



06

Data Storage: Postgres

Data Storage: Postgres

Overview

The following tasks were carried out to setup data storage in Postgres for source and transformed data:

1. This is the setup of schemas in Postgres. The following schemas and their purpose is shown below:

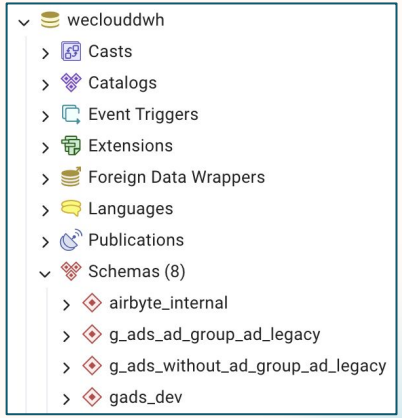
airbyte_internal: automatically created by airbyte

g_ads_ad_group_ad_legacy: Google Ads raw source data for table called Ad Group Ad Legacy ingested by Airbyte is stored in this schema

g_ads_without_ad_group_ad_legacy: All Google Ads raw source data for all tables except for Ad Group Ad Legacy ingested by Airbyte are stored in this schema (see slide 22 for list of tables)

gads_dev: All output of dbt models are stored in this schema

In the **next slide**, we will discuss about **gads_dev** schema in more detail

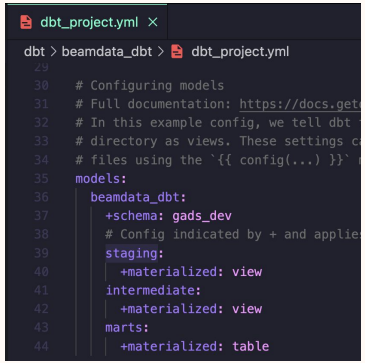


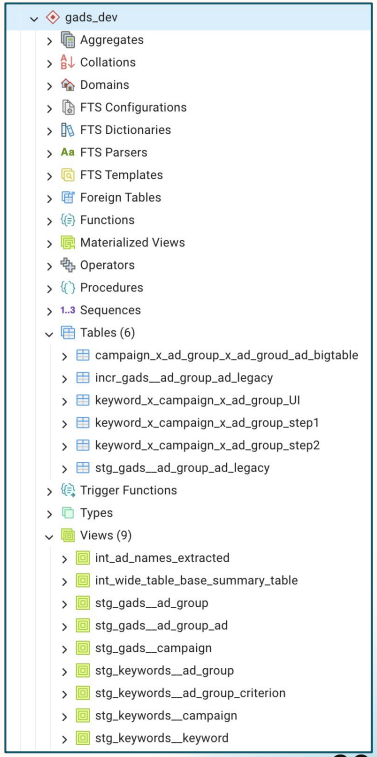
Data Storage: Postgres

To discuss more about the **gads_dev** schema that is populated by dbt models

1. **In dbt** as discussed in Section 5, the ELT process used in this project is a multi-layered data architecture called "staging-intermediate-mart". There are **3 layers**; Staging, Intermediate and Mart Layer. As shown in the figure below here are the configurations for each layer, all dbt models' output is stored in **gads_dev schema**:

   - **Staging:** models in this layer is outputted as a **view**
   - **Intermediate:** models in this layer is outputted as a **view**
   - **Mart:** models in this layer is outputted as a **table**





As shown in the figure on the far right, most of the output matches the rules mentioned above with the exception of the following **incr_gads__ad_group_ad_legacy** and **stg_gads__ad_group_ad_legacy** this is due to Postgres limitation for incremental models the base table which is **stg_gads__ad_group_ad_legacy** has to be a table and not a view. In addition, as per dbt the incremental model has to be stored as a table resulting in these 2 exceptions

Data Storage: Postgres

To discuss more about the debugging of error found in the sublayer of **Keyword Mart Layer** model as discussed in Section 5: Data Transformation: dbt slide 24

As discussed in slide 24 a solution to solve the following error was to break the model in multiple models, after which we encountered another similar error as shown below. A solution for this repeated error was to increase docker container's size of the shared memory directory to 1gb. The modification is shown on line 5 of the docker compose code shown in the figure on the right side:

crondbtlogfile.log

```
425  [0m23:30:07  9 of 15 OK created sql view model gads_dev.int_ad_names_extracted ..... [32mCREATE VIEW[0m in 0.05s]
426  [0m23:30:07  10 of 15 START sql incremental model gads_dev.incr_gads__ad_group_ad_legacy .... [RUN]
427  [0m23:30:07  10 of 15 OK created sql incremental model gads_dev.incr_gads__ad_group_ad_legacy  [32mINSERT 0 2226[0m in 0.19s]
428  [0m23:30:07  11 of 15 START sql view model gads_dev.int_wide_table_base_summary_table ..... [RUN]
429  [0m23:30:07  11 of 15 OK created sql view model gads_dev.int_wide_table_base_summary_table .. [32mCREATE VIEW[0m in 0.06s]
430  [0m23:30:07  12 of 15 START sql table model gads_dev.keyword_x_campaign_x_ad_group_step1 .... [RUN]
431  [0m23:33:26  12 of 15 OK created sql table model gads_dev.keyword_x_campaign_x_ad_group_step1  [32mSELECT 1015223[0m in 198.78s]
432  [0m23:33:26  13 of 15 START sql table model gads_dev.campaign_x_ad_group_x_ad_groud_ad_bigtable  [RUN]
433  [0m23:33:40  13 of 15 OK created sql table model gads_dev.campaign_x_ad_group_x_ad_groud_ad_bigtable  [32mSELECT 20302[0m in 14.09s]
434  [0m23:33:40  14 of 15 START sql table model gads_dev.keyword_x_campaign_x_ad_group_step2 .... [RUN]
435  [0m23:33:41  14 of 15 ERROR creating sql table model gads_dev.keyword_x_campaign_x_ad_group_step2  [31mERROR[0m in 0.27s]
436  [0m23:33:41  15 of 15 SKIP relation gads_dev.keyword_x_campaign_x_ad_group_UI ..... [33mSKIP[0m]
437  [0m23:33:41
438  [0m23:33:41  Finished running 9 view models, 5 table models, 1 incremental model in 0 hours 3 minutes and 34.38 seconds (214.38s).
439  [0m23:33:41
440  [0m23:33:41  [31mCompleted with 1 error and 0 warnings:[0m
441  [0m23:33:41
442  [0m23:33:41    Database Error in model keyword_x_campaign_x_ad_group_step2 (models/marts/keyword/keyword_x_campaign_x_ad_group_step2.sql)
443    could not resize shared memory segment "/PostgreSQL.1429225626" to 16777216 bytes: No space left on device
444    compiled Code at target/run/beamdata_dbt/models/marts/keyword/keyword_x_campaign_x_ad_group_step2.sql
445  [0m23:33:41
446  [0m23:33:41  Done. PASS=13 WARN=0 ERROR=1 SKIP=1 TOTAL=15
```

postgres > docker-compose.yml

```
1   version: '3.8'
2   services:
3     db:
4       image: postgres:latest
5       shm_size: 1gb
6       restart: unless-stopped
7       environment:
8         POSTGRES_DB: '[redacted]my_database_name
9         POSTGRES_USER: [redacted]
10        POSTGRES_PASSWORD: [redacted]
11      volumes:
12        - my_dbdata:/var/lib/postgresql/data
13      ports:
14        - "5432:5432"
15  volumes:
16    my_dbdata:
```

07

Visualization: Looker Studio

Visualization: Looker Studio

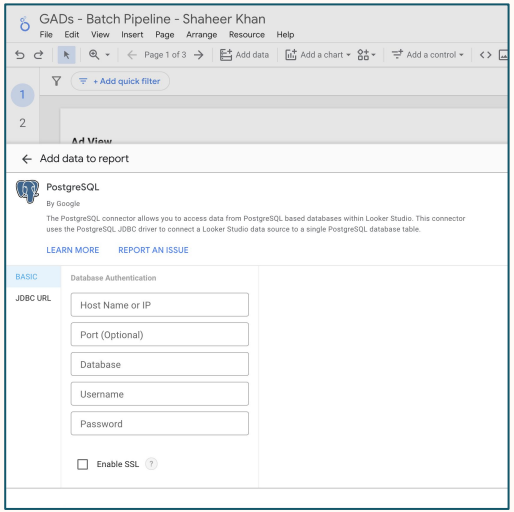
Overview

The following tasks were carried out to connect **Ads and Keyword Data Marts** created in **Postgres** with **Looker Studio**

Looker Studio was chosen as the **data visualization tool** for its ease of use and streamlined connectivity with Postgres. The data marts (big table) for Ads and Keyword view were created to depict the Google Ads user interface which is shown in the Looker Studio report

1. To setup connection between **Looker Studio** and **Postgres** input the following as shown below:

    - Host Name or IP
    - Port
    - Database Name
    - Username
    - Password



In the **next slide**, we will discuss more about Looker Studio and Postgres connection

Visualization: Looker Studio

To continue with the connection process of **Ads and Keyword Data Marts** created in **Postgres** with **Looker Studio**

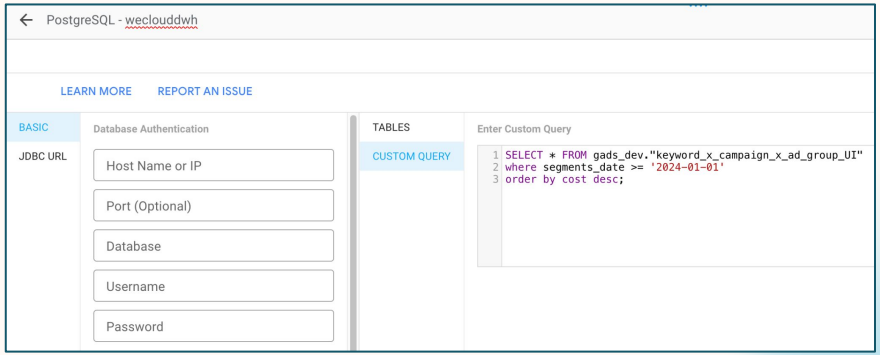
2. Next, to conclude the connection process between Looker Studio and Postgres for the Data Marts we would need to use custom query to connect with the correct schema and Data Mart tables, as shown below:

For Ad View: This has the **Big Table** depicting the **Ad View** in Google Ads website. The following query is used in Looker Studio when connecting with this table

```
SELECT * FROM gads_dev.campaign_x_ad_group_x_ad_groud_ad_bigtable
where segments_date >= '2024-01-01'
order by cost desc;
```

For Keyword View: This has the **Big Table** depicting the **Keyword View** in Google Ads website. The following query is used in Looker Studio when connecting with this table

```
SELECT * FROM gads_dev."keyword_x_campaign_x_ad_group_UI"
where segments_date >= '2024-01-01'
order by cost desc;
```



In the **next slide**, we will compare data in Looker Studio and Google Ads user interface

Visualization: Looker Studio

Now we will **compare data** in **Looker Studio** and **Google Ads user interface** to verify the accuracy of our pipeline for **Ad View**

Looker Studio Ad View: This has the Big Table depicting the **Ad View** in Google Ads website. Due to the conversion window and campaign launches timeline, **Ad** data in Looker Studio will match Google Ads user interface only for the **date range of greater than or equals to last 41 days**

GADs - Batch Pipeline - Shaheer Khan

Reset | Share | Edit

1
2

Ad View

Oct 21, 2024 - Oct 21, 2024

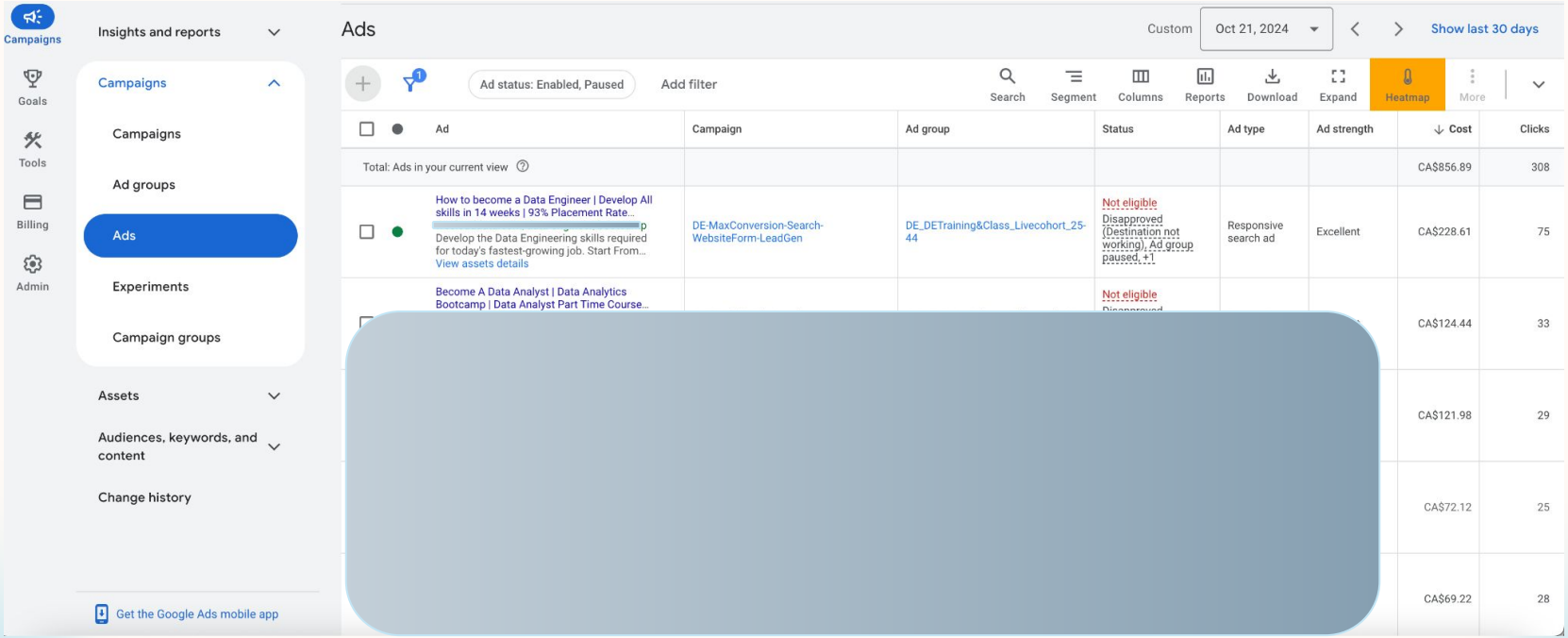
	segments_date	ad_group_ad_ad_name_1	ad_group_ad_ad_name_2	ad_group_ad_ad_name_3	ad_group_ad_ad_name_4	campaign_name	ad_group_name	ad_group_ad_status	cost	clicks
1.	Oct 21, 2024	How to become a Data Engineer	Develop All skills in 14 weeks	{LOCATION(City)}	Data Engineering Bootcamp 2024	DE-MaxConversion-Search-WebsiteForm-LeadGen	DE_DETraining&Class_Livecohort_25-44	ENABLED	228	75
2.	Oct 21, 2024							ENABLED	124	33
3.	Oct 21, 2024							ENABLED	121	29
4.	Oct 21, 2024							PAUSED	72	25
5.	Oct 21, 2024							ENABLED	69	28

In the **next slide**, we will compare this **Ad View** data shown above in **Looker Studio** with the data found for the same date range in **Google Ads user interface**

Visualization: Looker Studio

Here is the **Google Ads user interface view** for **Ad View**

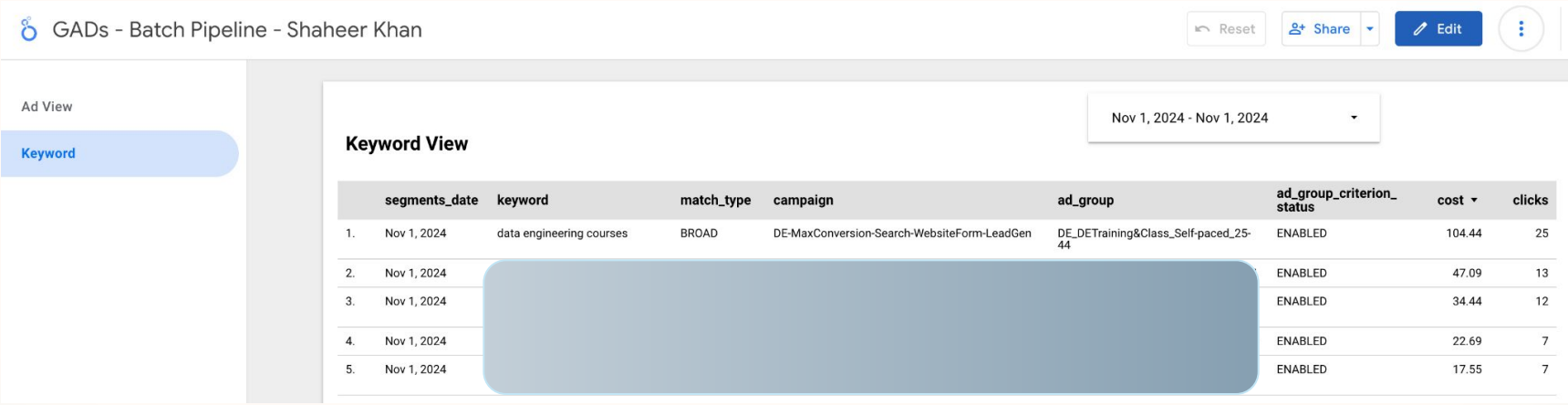
Google Ads Ad View: As shown below the data for Ad View matches the output from the pipeline in Looker Studio



Visualization: Looker Studio

Now we will **compare data** in **Looker Studio** and **Google Ads user interface** to verify the accuracy of our pipeline for **Keyword View**

Looker Studio Keyword View: This has the Big Table depicting the **Keyword View** in Google Ads website. Due to the conversion window and campaign launches timeline, **Keyword** data in Looker Studio will match Google Ads user interface only for the **date range of greater than or equals to last 30 days**



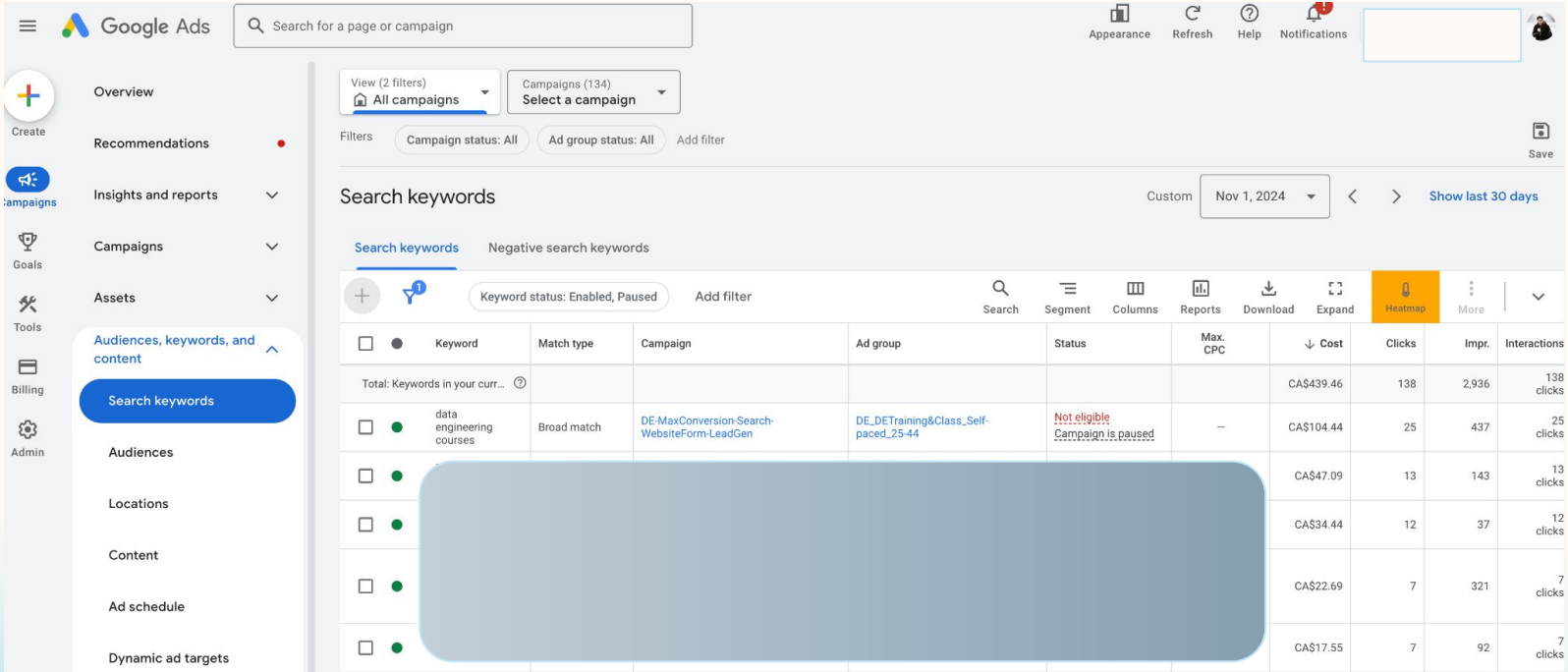
	segments_date	keyword	match_type	campaign	ad_group	ad_group_criterion_status	cost	clicks
1.	Nov 1, 2024	data engineering courses	BROAD	DE-MaxConversion-Search-WebsiteForm-LeadGen	DE_DETraining&Class_Self-paced_25-44	ENABLED	104.44	25
2.	Nov 1, 2024					ENABLED	47.09	13
3.	Nov 1, 2024					ENABLED	34.44	12
4.	Nov 1, 2024					ENABLED	22.69	7
5.	Nov 1, 2024					ENABLED	17.55	7

In the **next slide**, we will compare this **Keyword View** data shown above in **Looker Studio** with the data found for the same date range in **Google Ads user interface**

Visualization: Looker Studio

Here is the **Google Ads user interface view** for **Keyword View**

Google Ads Keyword View: As shown below the data for Keyword View matches the output from the pipeline in Looker Studio



08

Future Iteration

Future Iteration

Overview

We will discuss improvements and new features that are to be added in the future iterations of this pipeline

1. **Incremental Loading:** For efficiency, especially with large datasets, incremental models load only the new or changed data. As one of the models was tested with incremental loading, for next iteration more models will be used with incremental loading. This would be implemented with data quality test to make sure accurate data is reported

2. **Orchestration:** incorporating orchestration tools such as Airflow, Prefect or migrating to dbt cloud to use their orchestration features could be added in place of crontab that is being used currently to deploy dbt models

3. **CI/CD:** integration of CI/CD brings many benefits few of which are; automation of the building, testing, and deployment of dbt project, secondly it allows for faster and more frequent deployment, thirdly improved quality so that errors are caught early through automated testing and lastly better collaboration with other data engineers and analyst. GitHub actions or dbt cloud would be used for CI/CD

In the **next slide**, we will discuss more about future improvements and features that are to be added

Future Iteration

4. **Data Validation:** to maintain high data quality in ingestion and transformation so that accurate and consistent data is reported on. This could be implemented using dbt tests (e.g., not_null, unique, accepted_values) to validate data quality after transformations.

   Another option for data validation is the incorporation of an open-source library called Great Expectations

5. **Error Handling and Logging:** to set up alerts and logging for errors, airbyte and dbt both have this functionality, for dbt specifically logging features and error handling mechanisms such as (e.g., on-run-end hooks) can be used

6. **Upgradation and Migration of Project:** to migrate this project from AWS to GCP to compare cost and features, incorporating Big Query instead of Postgres

Do you have any questions?

Email: shaheer.ak.200@gmail.com
Linkedin: https://www.linkedin.com/in/shaheerkhan200/
Website: https://shaheerkhanalytics.com/

Thanks!