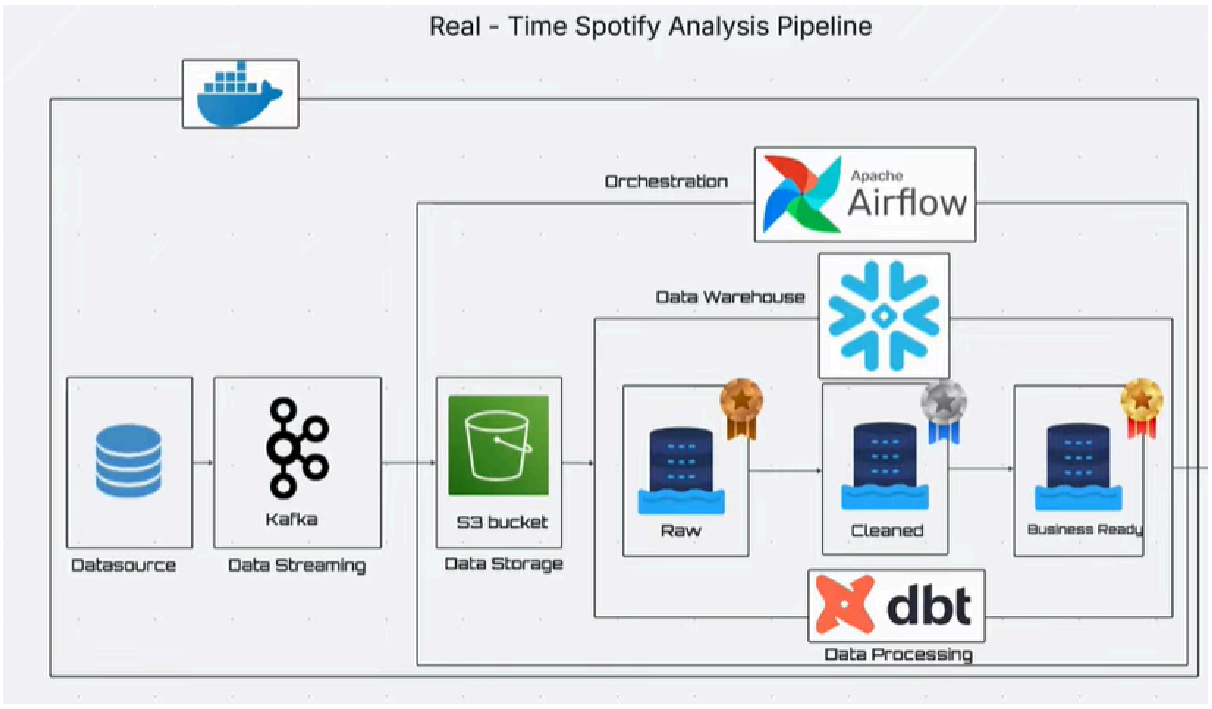
SPOTIFY DATA PIPELINE

Architecture:



Tech-Stack Used:
1.Kafka
2.Snowflake
3.Airflow
4.Dbt
5.Docker
6.AWS S3

Development
1. Create a dir simulator (mkdir simulator), this will contain the producer code that will produce data in real-time.
2.Create docker dir which will have a docker-compose file having all the containers: zookeeper, kafka, kafdrop(for UI), minio for storage, postgres for initializing airflow & airflow containers. We will also create a .env file.(Make your credentials)
3. Create the requirement.txt file and install all dependencies:
pip install -r requirements.txt # here -r is for read
4. Do docker compose up after changing dir to docker.
5. Check if kafdrop, airflow and minio are working by clicking the port in docker desktop. Check login creds as well.
6. Now let’s come to producer code:

6.1 We set up Kafka config and created a Kafka Producer with those config and a value serializer that will convert the dictionary to json.

6.2 Then we created a song_artist_pair list of dict, & for each artist-song pair we generate a uuid(universally unique identifier) [uuid5: `same song_id each time`]. A **namespace** is just a fixed UUID used as a *seed* to generate other UUIDs in a consistent way.

Key properties of uuid5:

- ✅ **Deterministic** (same input → same UUID every time)
- ❌ **Not random**
- ✅ Ideal for **natural keys** → **surrogate IDs**

uuid4 is used to create “random” uuid

6.3 Then we generated random users, and created a “generate_event” function to generate random records using the above data.

7. **Kafka Consumer**: We will first connect to s3 service and check if a bucket already exists, else we create a new one.

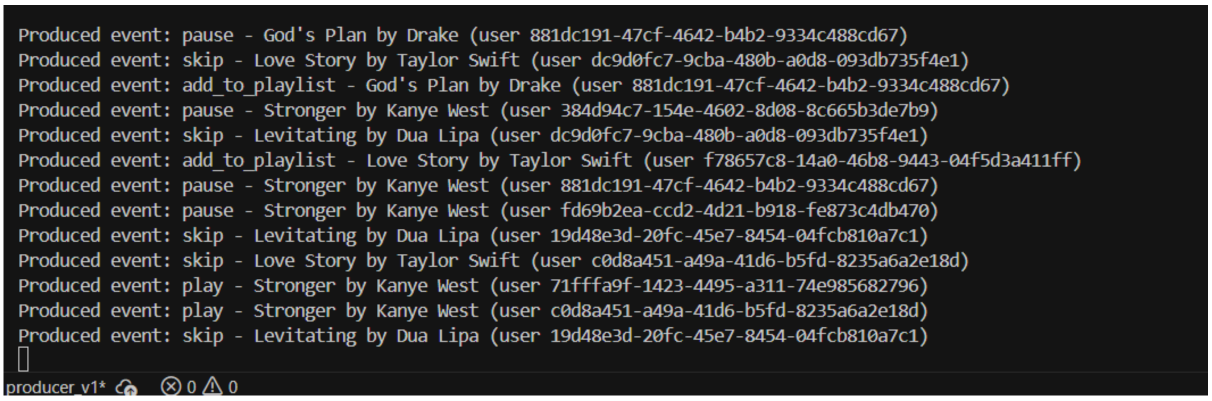
7.1 Initialize a Kafka Consumer and store the events in s3 bucket according to batch size.

7.2 auto_offset_reset="earliest” means it will continue from where it last stopped

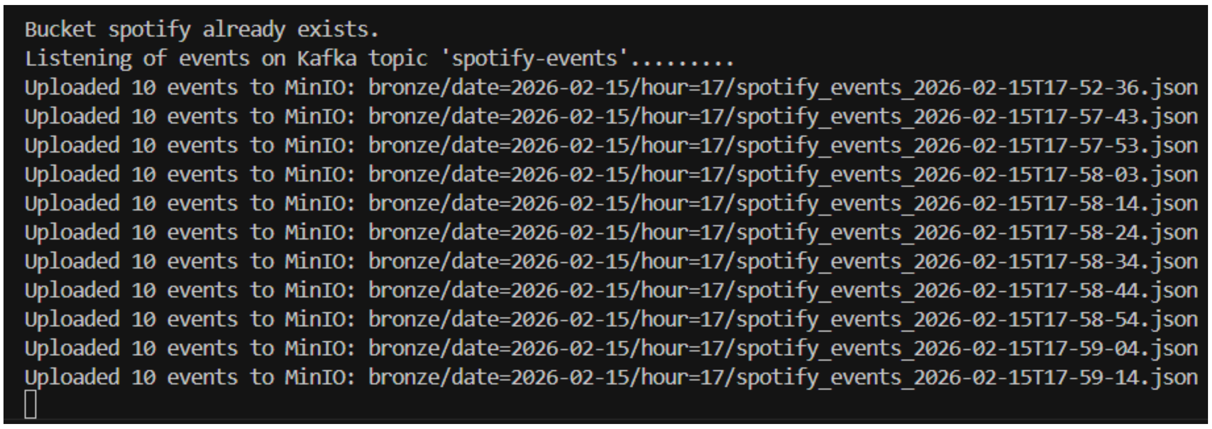
7.3 kafka_group_id - It saves the offset , if we change this whole offset will start again

7.4 json.load will convert json to dict, take a string as input and returns a dictionary as output.

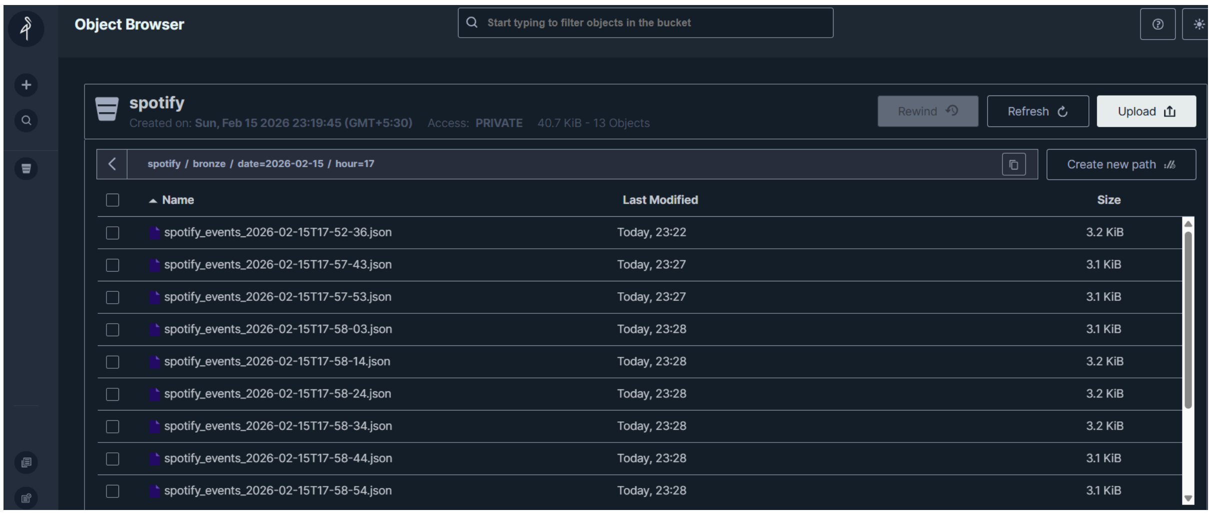
7.5 json.dumps will convert dict to json, take a dictionary as input and returns a string as output.



```
Produced event: pause - God's Plan by Drake (user 881dc191-47cf-4642-b4b2-9334c488cd67)
Produced event: skip - Love Story by Taylor Swift (user dc9d0fc7-9cba-480b-a0d8-093db735f4e1)
Produced event: add_to_playlist - God's Plan by Drake (user 881dc191-47cf-4642-b4b2-9334c488cd67)
Produced event: pause - Stronger by Kanye West (user 384d94c7-154e-4602-8d08-8c665b3de7b9)
Produced event: skip - Levitating by Dua Lipa (user dc9d0fc7-9cba-480b-a0d8-093db735f4e1)
Produced event: add_to_playlist - Love Story by Taylor Swift (user f78657c8-14a0-46b8-9443-04f5d3a411ff)
Produced event: pause - Stronger by Kanye West (user 881dc191-47cf-4642-b4b2-9334c488cd67)
Produced event: pause - Stronger by Kanye West (user fd69b2ea-ccd2-4d21-b918-fe873c4db470)
Produced event: skip - Levitating by Dua Lipa (user 19d48e3d-20fc-45e7-8454-04fcb810a7c1)
Produced event: skip - Love Story by Taylor Swift (user c0d8a451-a49a-41d6-b5fd-8235a6a2e18d)
Produced event: play - Stronger by Kanye West (user 71fffa9f-1423-4495-a311-74e985682796)
Produced event: play - Stronger by Kanye West (user c0d8a451-a49a-41d6-b5fd-8235a6a2e18d)
Produced event: skip - Levitating by Dua Lipa (user 19d48e3d-20fc-45e7-8454-04fcb810a7c1)
```



```
Bucket spotify already exists.
Listening of events on Kafka topic 'spotify-events'.........
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-52-36.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-57-43.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-57-53.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-58-03.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-58-14.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-58-24.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-58-34.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-58-44.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-58-54.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-59-04.json
Uploaded 10 events to MinIO: bronze/date=2026-02-15/hour=17/spotify_events_2026-02-15T17-59-14.json
```



spotify
Created on: Sun, Feb 15 2026 23:19:45 (GMT+5:30) Access: PRIVATE 40.7 KiB - 13 Objects

spotify / bronze / date=2026-02-15 / hour=17

Name	Last Modified	Size
spotify_events_2026-02-15T17-52-36.json	Today, 23:22	3.2 KiB
spotify_events_2026-02-15T17-57-43.json	Today, 23:27	3.1 KiB
spotify_events_2026-02-15T17-57-53.json	Today, 23:27	3.1 KiB
spotify_events_2026-02-15T17-58-03.json	Today, 23:28	3.1 KiB
spotify_events_2026-02-15T17-58-14.json	Today, 23:28	3.2 KiB
spotify_events_2026-02-15T17-58-24.json	Today, 23:28	3.2 KiB
spotify_events_2026-02-15T17-58-34.json	Today, 23:28	3.2 KiB
spotify_events_2026-02-15T17-58-44.json	Today, 23:28	3.1 KiB
spotify_events_2026-02-15T17-58-54.json	Today, 23:28	3.1 KiB

Now we have data in s3 bucket

8. Move data from S3 to Snowflake using Airflow:

8.1 Create a DAG (spotify_minio_to_snowflake_bronze)

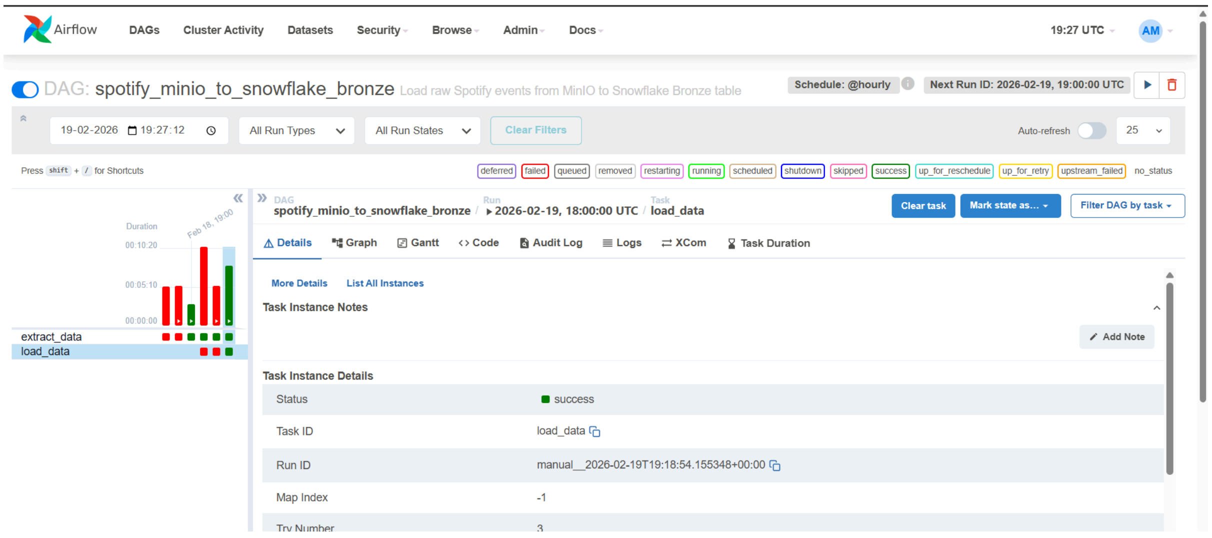
8.2 Two tasks : extract_task to extract data from s3, and load_task to load data in snowflake bronze layer.

8.3 Extract task will extract json files data and store it into temp file on local path

8.4 load_to_snowflake will load raw data in the SF bronze layer for further processing.

**context : Airflow context dictionary that contains metadata and information about a running Directed Acyclic Graph (DAG), its tasks, and the Airflow environment.

Below is the successful run of Airflow DAG which loaded data into snowflake DB bronze layer table:



DAG: spotify_minio_to_snowflake_bronze Load raw Spotify events from MinIO to Snowflake Bronze table

Schedule: @hourly

Next Run ID: 2026-02-19, 19:00:00 UTC

19-02-2026 19:27:12 | All Run Types | All Run States | Clear Filters | Auto-refresh | 25

Press shift + / for Shortcuts

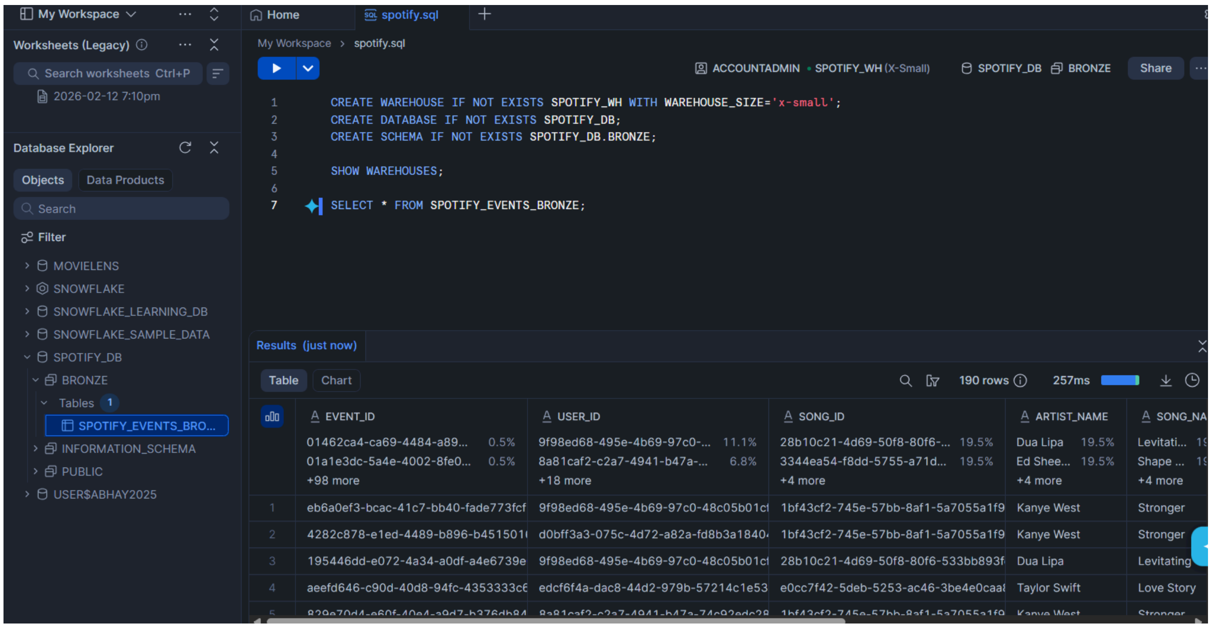
deferred failed queued removed restarting running scheduled shutdown skipped success up_for_reschedule up_for_retry upstream_failed no_status

DAG spotify_minio_to_snowflake_bronze / Run 2026-02-19, 18:00:00 UTC / Task load_data

Task Instance Notes

Task Instance Details

Status	🟩 success
Task ID	load_data
Run ID	manual__2026-02-19T19:18:54.155348+00:00
Map Index	-1
Try Number	3



```sql
CREATE WAREHOUSE IF NOT EXISTS SPOTIFY_WH WITH WAREHOUSE_SIZE='x-small';
CREATE DATABASE IF NOT EXISTS SPOTIFY_DB;
CREATE SCHEMA IF NOT EXISTS SPOTIFY_DB.BRONZE;

SHOW WAREHOUSES;

SELECT * FROM SPOTIFY_EVENTS_BRONZE;
```

Results (just now)

190 rows 257ms

	EVENT_ID	USER_ID	SONG_ID	ARTIST_NAME	SONG_NA
	01462ca4-ca69-4484-a89... 0.5% 01a1e3dc-5a4e-4002-8fe0... 0.5% +98 more	9f98ed68-495e-4b69-97c0-... 11.1% 8a81caf2-c2a7-4941-b47a-... 6.8% +18 more	28b10c21-4d69-50f8-80f6-... 19.5% 3344ea54-f8dd-5755-a71d... 19.5% +4 more	Dua Lipa 19.5% Ed Shee... 19.5% +4 more	Levitati... Shape ... +4 more
1	eb6a0ef3-bcac-41c7-bb40-fade773fcf	9f98ed68-495e-4b69-97c0-48c05b01c	1bf43cf2-745e-57bb-8af1-5a7055a1f9	Kanye West	Stronger
2	4282c878-e1ed-4489-b896-b451501	d0bff3a3-075c-4d72-a82a-fd8b3a1840	1bf43cf2-745e-57bb-8af1-5a7055a1f9	Kanye West	Stronger
3	195446dd-e072-4a34-a0df-a4e6739e	9f98ed68-495e-4b69-97c0-48c05b01c	28b10c21-4d69-50f8-80f6-533bb893f	Dua Lipa	Levitating
4	aeefd646-c90d-40d8-94fc-4353333c6	edcf6f4a-dac8-44d2-979b-57214c1e53	e0cc7f42-5deb-5253-ac46-3be4e0caa	Taylor Swift	Love Story
5	829e70d4-e60f-40e4-a9d7-b376db84	8a81caf2-c2a7-4941-b47a-74c92edc28	1bf43cf2-745e-57bb-8af1-5a7055a1f9	Kanye West	Stronger

9. DBT:

9.1 Create a new folder spotify_dbt

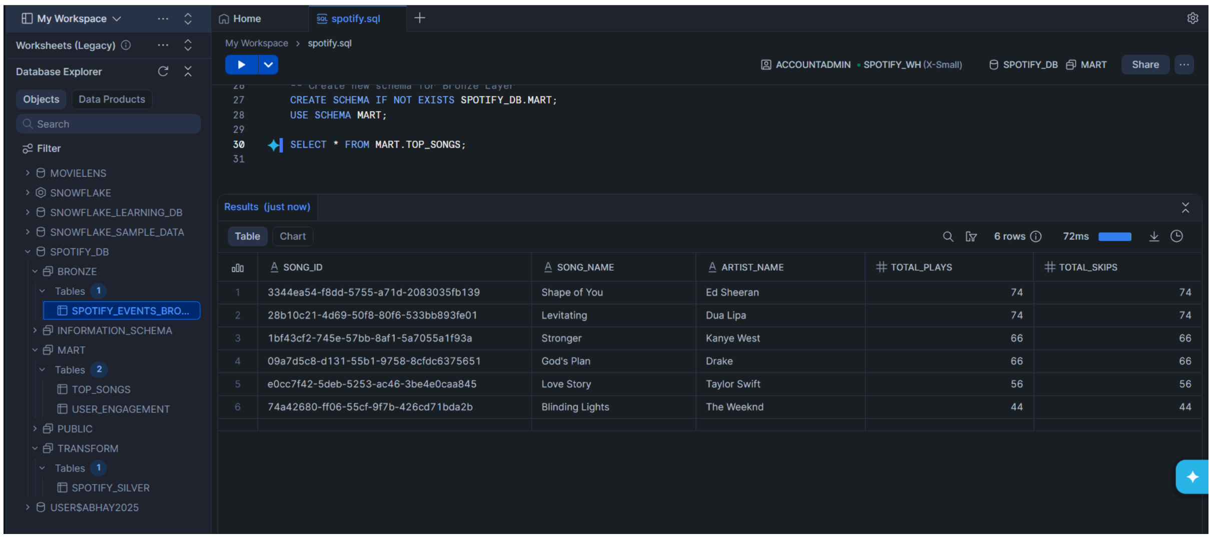
9.2 Initialize dbt: dbt init and enter the details of snowflake account , warehouse, database, default schema(we used Transform schema created as default), threads, it will create a profile.

9.3 Create sources.yml file containing metadata for silver layer source.

9.4 Create silver dir & spotify_sql table to transform data from bronze layer and store in silver layer.

9.5 Similarly create models for the gold layer.

9.6 Run >> dbt run inside spotify_dbt dir

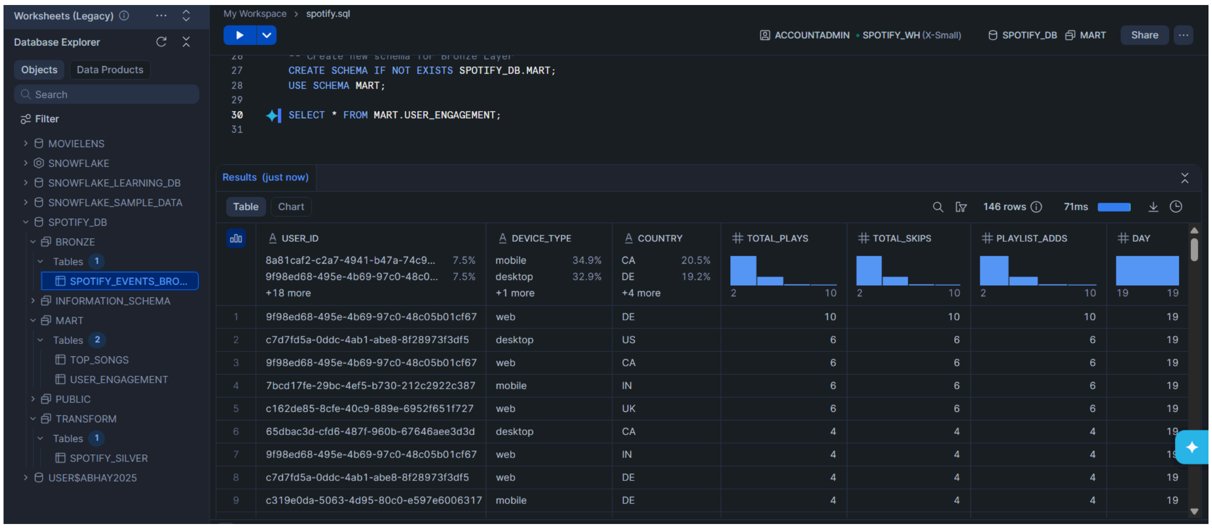


```sql
-- Create new schema for Bronze Layer
CREATE SCHEMA IF NOT EXISTS SPOTIFY_DB.MART;
USE SCHEMA MART;

SELECT * FROM MART.TOP_SONGS;

```

	SONG_ID	SONG_NAME	ARTIST_NAME	TOTAL_PLAYS	TOTAL_SKIPS
1	3344ea54-f8dd-5755-a71d-2083035fb139	Shape of You	Ed Sheeran	74	74
2	28b10c21-4d69-50f8-80f6-533bb893fe01	Levitating	Dua Lipa	74	74
3	1bf43cf2-745e-57bb-8af1-5a7055a1f93a	Stronger	Kanye West	66	66
4	09a7d5c8-d131-55b1-9758-8cfdc6375651	God's Plan	Drake	66	66
5	e0cc7f42-5deb-5253-ac46-3be4e0caa845	Love Story	Taylor Swift	56	56
6	74a42680-ff06-55cf-9f7b-426cd71bda2b	Blinding Lights	The Weeknd	44	44



```sql
-- Create new schema for Bronze Layer
CREATE SCHEMA IF NOT EXISTS SPOTIFY_DB.MART;
USE SCHEMA MART;

SELECT * FROM MART.USER_ENGAGEMENT;

```

	USER_ID	DEVICE_TYPE	COUNTRY	TOTAL_PLAYS	TOTAL_SKIPS	PLAYLIST_ADDS	DAY
	8a81caf2-c2a7-4941-b47a-74c9... 7.5% 9f98ed68-495e-4b69-97c0-48c0... 7.5% +18 more	mobile 34.9% desktop 32.9% +1 more	CA 20.5% DE 19.2% +4 more	2 10	2 10	2 10	19 19
1	9f98ed68-495e-4b69-97c0-48c05b01cf67	web	DE	10	10	10	19
2	c7d7fd5a-0ddc-4ab1-abe8-8f28973f3df5	desktop	US	6	6	6	19
3	9f98ed68-495e-4b69-97c0-48c05b01cf67	web	CA	6	6	6	19
4	7bcd17fe-29bc-4ef5-b730-212c2922c387	mobile	IN	6	6	6	19
5	c162de85-8cfe-40c9-889e-6952f651f727	web	UK	6	6	6	19
6	65dbac3d-cfd6-487f-960b-67646aee3d3d	desktop	CA	4	4	4	19
7	9f98ed68-495e-4b69-97c0-48c05b01cf67	web	IN	4	4	4	19
8	c7d7fd5a-0ddc-4ab1-abe8-8f28973f3df5	web	DE	4	4	4	19
9	c319e0da-5063-4d95-80c0-e597e6006317	mobile	DE	4	4	4	19

Note: Please make sure to update .env files with correct username, password and account of snowflake before running.

Thanks for reading the documentation.