HW 3.2: dbt, Airflow, and Snowflake

Group 5:

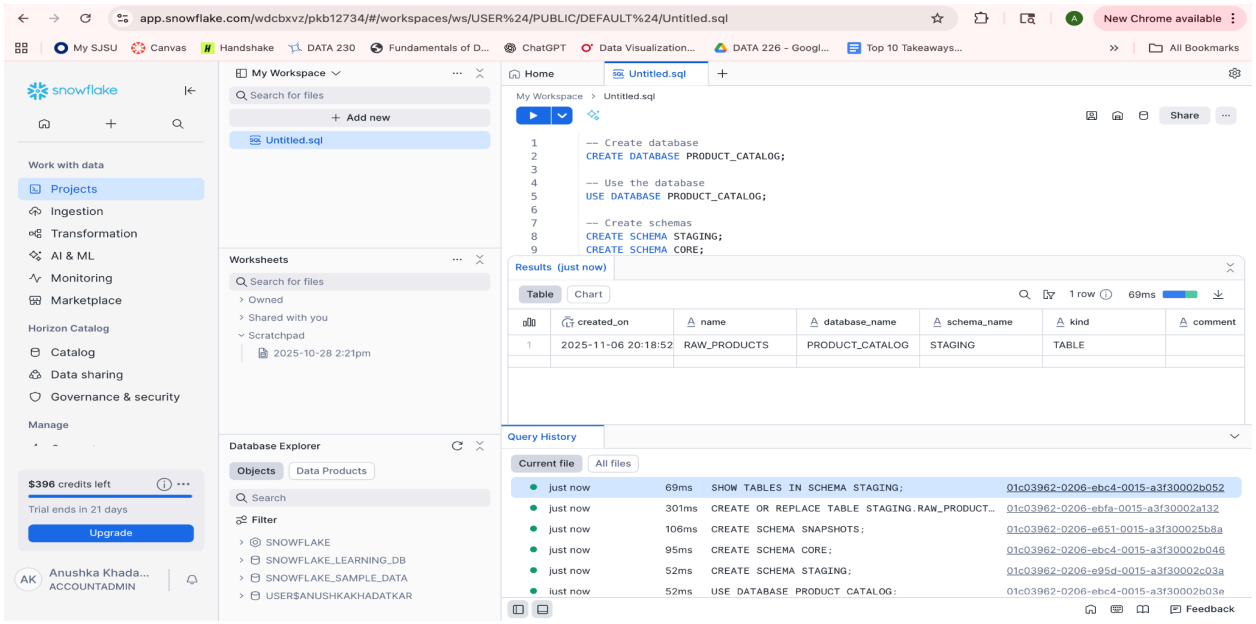
- Anushka Rajesh Khadatkar – 018383963
- Kanika Mamgain – 018319704
- Samruddhi Suresh Chitnis – 018452122
- Soroor Ghandali – 018286281

---

1. Set up the development environment:
   Successfully installed Python, Apache Airflow, DBT, and the snowflake connector.

```
simmi@Anushkas-MacBook-Air ~ % source ~/hw3.2_pipeline/venv/bin/activate
(venv) simmi@Anushkas-MacBook-Air ~ % pip list | grep -E "airflow|dbt|snowflake"
apache-airflow                          2.9.3
apache-airflow-providers-common-io      1.3.2
apache-airflow-providers-common-sql     1.14.2
apache-airflow-providers-fab            1.2.2
apache-airflow-providers-ftp            3.10.0
apache-airflow-providers-http           4.12.0
apache-airflow-providers-imap           3.6.1
apache-airflow-providers-smtp           1.7.1
apache-airflow-providers-snowflake      5.6.0
apache-airflow-providers-sqlite         3.8.1
dbt-adapters                            1.9.0
dbt-common                              1.12.0
dbt-core                                1.8.7
dbt-extractor                           0.6.0
dbt-semantic-interfaces                 0.5.1
dbt-snowflake                           1.8.3
snowflake-connector-python              3.11.0
snowflake-sqlalchemy                    1.6.1
(venv) simmi@Anushkas-MacBook-Air ~ % python -V
Python 3.11.14
(venv) simmi@Anushkas-MacBook-Air ~ % dbt --version
Core:
  - installed: 1.8.7
  - latest:    1.10.13 - Update available!

  Your version of dbt-core is out of date!
  You can find instructions for upgrading here:
  https://docs.getdbt.com/docs/installation

Plugins:
  - snowflake: 1.8.3 - Update available!

  At least one plugin is out of date or incompatible with dbt-core.
  You can find instructions for upgrading here:
  https://docs.getdbt.com/docs/installation


(venv) simmi@Anushkas-MacBook-Air ~ % airflow version
2.9.3
```

Created a snowflake account.



2. Source a Dataset:

Selected Home Depot products dataset with 2,551 products stored in local filesystem.

Dataset link (“products_prepared.csv”) - https://drive.google.com/file/d/1cHH6MDRTEG32rh4iLxfz6JOVn-hlvgdW/view?usp=sharing

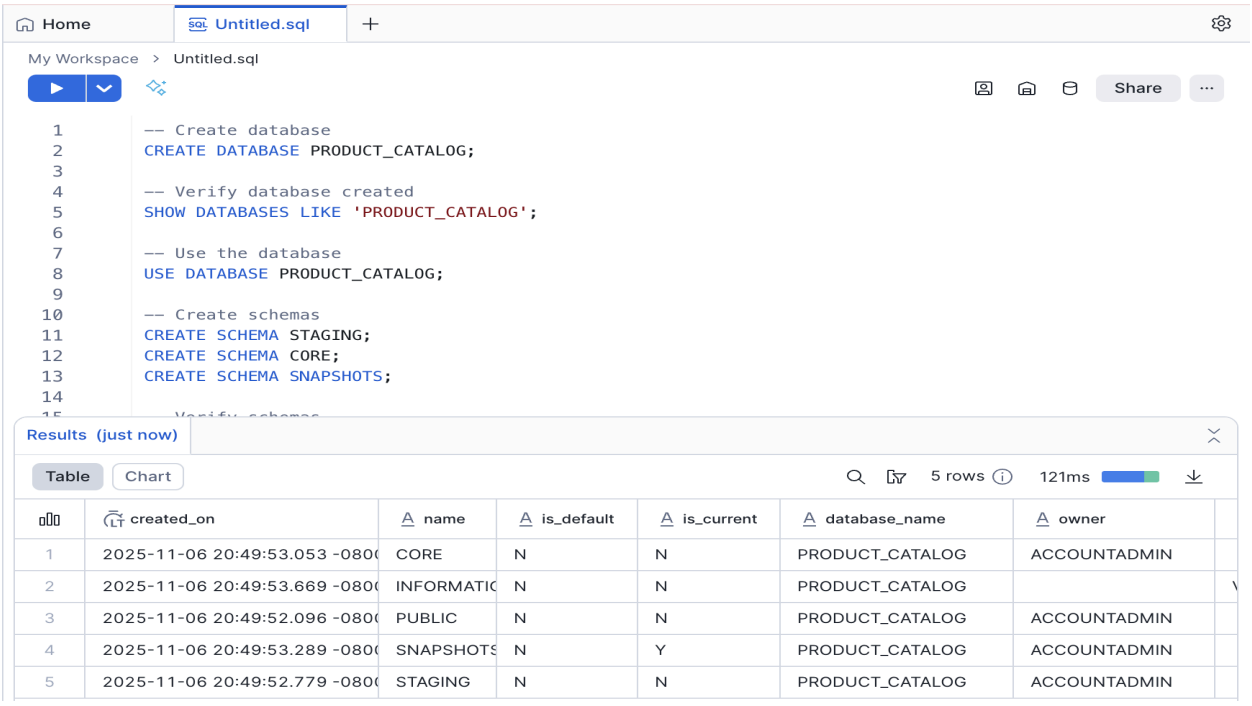
Top 10 rows of our dataset -

```
(data_pipeline_venv) simmi@Anushkas-MacBook-Air ~ % cd ~/data_pipeline_project
(data_pipeline_venv) simmi@Anushkas-MacBook-Air data_pipeline_project % ls -lh data/
total 3112
-rw-r--r--  1 simmi  staff   1.5M  6 Nov 15:28 products.csv
(data_pipeline_venv) simmi@Anushkas-MacBook-Air data_pipeline_project % head -10 data/products.csv
id,name,category,price,description,updated_at
310090686,Men's 3X Large Carbon Heather Cotton/Polyester Rain Defender Paxton Heavyweight Hooded Zip-Front Sweatshirt,Carhartt,64.99,"This heavyweight, water-repellent hooded sweatshirt has a zip front fo
r fast layering. ORIGINAL FIT. 13 oz., 75% cotton/25% polyester blend with Rain Defender durable water repellent. Attached, jersey-lined three-piece hood with drawcord closure. Antique-finish brass front
zipper. Two front hand-warmer pockets have a hidden security pocket inside. Stretchable, spandex-reinforced rib-knit cuffs and waistband. Locker loop facilitates hanging.",2021-12-14 00:55:53+0000
206724580,Turmode 30 ft. RP TNC Female to RP TNC Male Adapter Cable,Unbranded,71.61,"If you need more length between your existing wireless device and Hi-Gain Antenna, this is the product for you. It's co
mpatible with most Wi-Fi Antennas, so it is easy for you to extend your wireless network. Just replace your existing cable that runs between your wireless device and Antenna and you're ready to use your n
etwork with extended range.",2021-12-14 00:55:53+0000
310347105,Large Tapestry Bolster Bed,Carolina Pet Company,166.83,"Polyester cover resembling rich Italian tapestries wraps your pet in comfort and style. 100% recycled polyester high loft MemoryFiber fill
 keeps pets elevated off cold floors for relief on tired joints and pressure points. A great fit for pets up to 100 lbs. Zippered removable cover is machine washable in cold water only. To dry, use a deli
cate cycle or gentle setting with very low heat.",2021-12-14 00:55:53+0000
312338711,16-Gauge-Sinks Vessel Sink in White with Faucet,Unbranded,507.63,It features a rectangle shape. This vessel set is designed to be installed as a undermount vessel set. It is constructed with cer
amic. This vessel set comes with an enamel glaze finish in White color. It is designed for a deck mount faucet.,2021-12-14 00:55:53+0000
308561619,Men's Crazy Horse 9'' Logger Boot - Steel Toe - Black Size 10.5(W),Adtec,103.59,This 9 in. black full grain leather logger boot is Tough. With plain soft toe and rubber outsole it provides depen
dable traction with a comfortable fit. It is made with Goodyear Welt Construction so it is sturdy and secure.,2021-12-14 00:55:54+0000
312917264,Mariana 6 ft. Multi-Color 3-Panel Screen Divider,HomeRoots,550.24,"With robust structure and sophisticated canvas printing, this 3-panel screen is colorfully designed to be a great way to carve
out space in your room or anywhere around the home. Not only will this elegant piece give you the needed exclusivity but it will also add inspiring style and beauty to your decor. This screen creatively f
eatures canvas print of a different and complementary images on each side. With its lightweight construction, our screen will definitely fill your space with inspiring visual flair for many-Years to come.
 3-panel screen finished on both sides Canvas printing.",2021-12-14 00:55:54+0000
316216346,5 gal. #650C-2 Powdery Mist Semi-Gloss Interior Paint,BEHR PRO,118.0,"BEHR PRO i300 Semi-Gloss Interior Paint has a sleek, radiant sheen appearance. This professional quality latex paint has sup
erior hide and coverage, excellent sprayability, spray and back-roll, and superior touch-up. Use on properly prepared and primed drywall, concrete, masonry, wood and metal surfaces in both residential and
 commercial applications.",2021-12-14 00:55:54+0000
312450240,7/8 in. x 4-1/2 in. x 0.045 in. Metal and Stainless Cutting Wheel (50-Pack),DEWALT,44.97,DEWALT High Performance 0.045 in. Metal Cutting Wheels have a thin 0.045 in. cutting edge design for fast
 burr free cutting. They have a proprietary aluminum oxide grain combination for aggressive cutting action and the proprietary material mix ensures durable long life wheels. They have 2-full sheets of fib
erglass for durability and user safety. These wheels are ideal for high performance cutting in all types of ferrous metals and stainless steel.,2021-12-14 00:55:54+0000
300980025,Ring Gold Bar Cart,Titan Lighting,278.0,This Ring Bar Cart is sure to make a statement in any room. This carts features a metal frame in gold and the two shelves are made of mirror with an antiq
ue finish. We offer a wide variety of furniture and decorative accessories to meet all of your decor needs.,2021-12-14 00:55:55+0000
(data_pipeline_venv) simmi@Anushkas-MacBook-Air data_pipeline_project %
```

3. Setup Snowflake:

Created a dedicated database and schema in snowflake and also defined staging tables to store raw product data.

a. Create a database, and schema for the project -



Home | Untitled.sql

My Workspace > Untitled.sql

Share

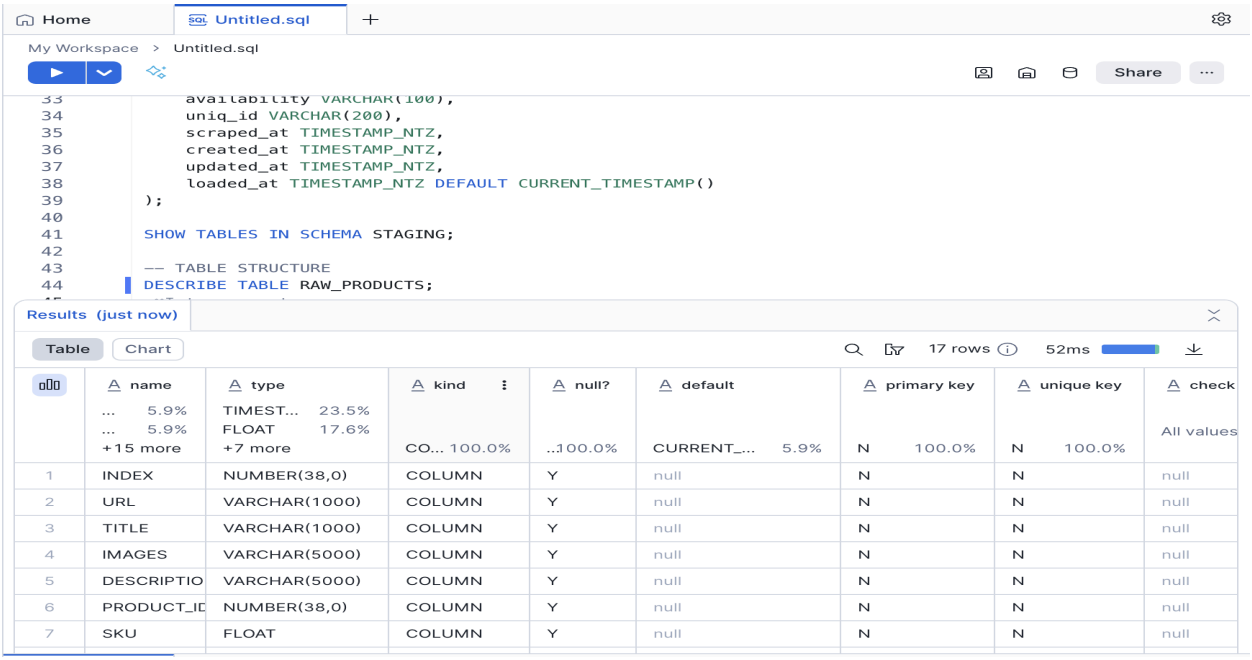
```
-- Create database
CREATE DATABASE PRODUCT_CATALOG;

-- Verify database created
SHOW DATABASES LIKE 'PRODUCT_CATALOG';

-- Use the database
USE DATABASE PRODUCT_CATALOG;

-- Create schemas
CREATE SCHEMA STAGING;
CREATE SCHEMA CORE;
CREATE SCHEMA SNAPSHOTS;
```

Results (just now)

Table | Chart — 5 rows — 121ms

	created_on	name	is_default	is_current	database_name	owner
1	2025-11-06 20:49:53.053 -080(	CORE	N	N	PRODUCT_CATALOG	ACCOUNTADMIN
2	2025-11-06 20:49:53.669 -080(	INFORMATIC	N	N	PRODUCT_CATALOG	
3	2025-11-06 20:49:52.096 -080(	PUBLIC	N	N	PRODUCT_CATALOG	ACCOUNTADMIN
4	2025-11-06 20:49:53.289 -080(	SNAPSHOTS	N	Y	PRODUCT_CATALOG	ACCOUNTADMIN
5	2025-11-06 20:49:52.779 -080(	STAGING	N	N	PRODUCT_CATALOG	ACCOUNTADMIN

b. Create a staging table to receive raw data from the source -



Home | Untitled.sql

My Workspace > Untitled.sql

Share

```
    availability VARCHAR(100),
    uniq_id VARCHAR(200),
    scraped_at TIMESTAMP_NTZ,
    created_at TIMESTAMP_NTZ,
    updated_at TIMESTAMP_NTZ,
    loaded_at TIMESTAMP_NTZ DEFAULT CURRENT_TIMESTAMP()
);

SHOW TABLES IN SCHEMA STAGING;

-- TABLE STRUCTURE
DESCRIBE TABLE RAW_PRODUCTS;
```

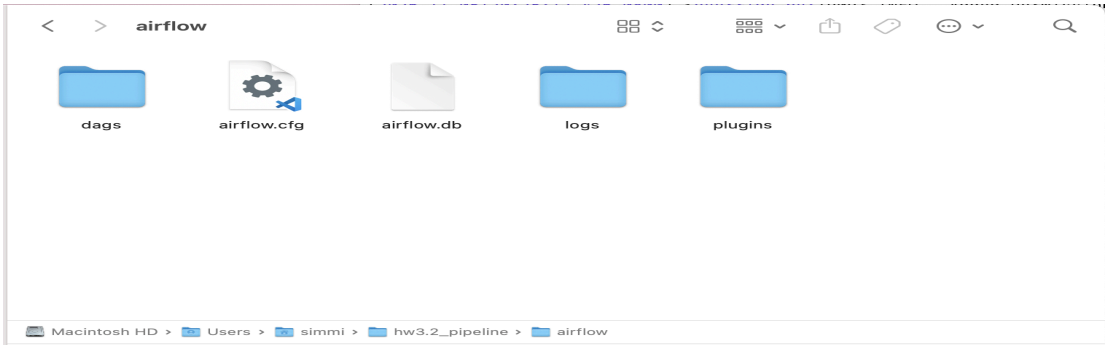
Results (just now)

Table | Chart — 17 rows — 52ms

	name	type	kind	null?	default	primary key	unique key	check
	... 5.9% ... 5.9% +15 more	TIMEST... 23.5% FLOAT 17.6% +7 more	CO... 100.0%	...100.0%	CURRENT_... 5.9%	N 100.0%	N 100.0%	All values
1	INDEX	NUMBER(38,0)	COLUMN	Y	null	N	N	null
2	URL	VARCHAR(1000)	COLUMN	Y	null	N	N	null
3	TITLE	VARCHAR(1000)	COLUMN	Y	null	N	N	null
4	IMAGES	VARCHAR(5000)	COLUMN	Y	null	N	N	null
5	DESCRIPTIO	VARCHAR(5000)	COLUMN	Y	null	N	N	null
6	PRODUCT_ID	NUMBER(38,0)	COLUMN	Y	null	N	N	null
7	SKU	FLOAT	COLUMN	Y	null	N	N	null

4. Implement Airflow DAG:

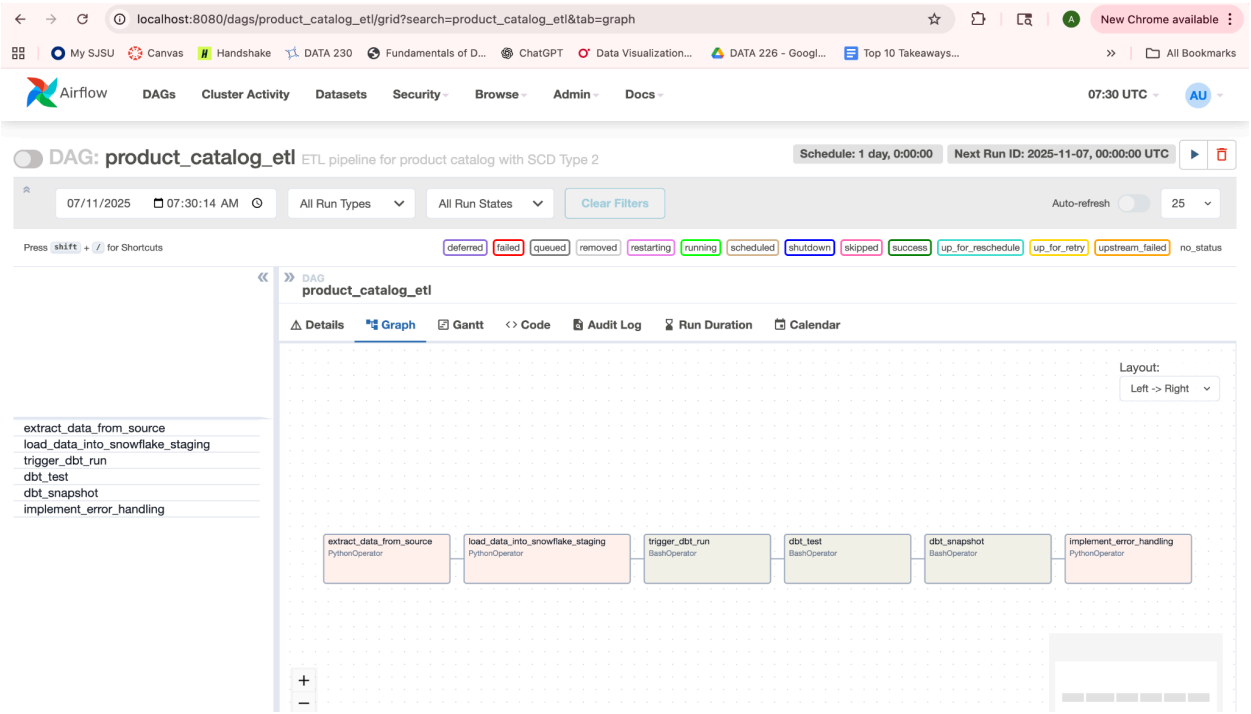
a. Create DAG file in the Airflow dags folder -



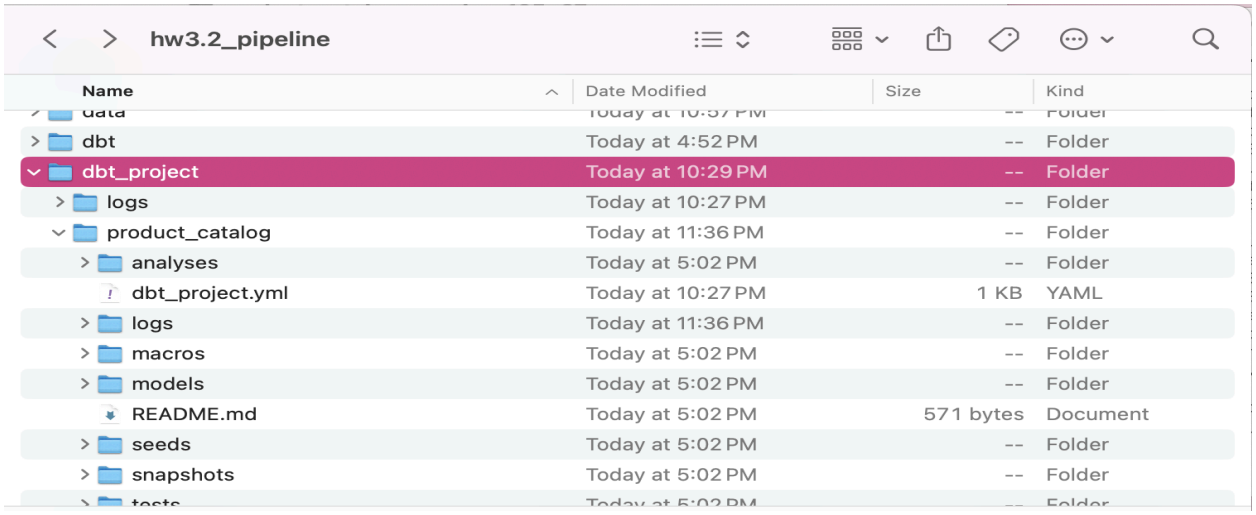
b. Tasks -

Built an airflow dag names “product_catalog_etl”

- Extracting data from the source: Fetch product catalog data from our source (i.e., local filesystem)
- Loading data into Snowflake staging table: loaded extracted data to the staging table in snowflake for processing in 6 batches with 500 rows per batch.
- Triggering dbt run: run dbt models to transform the data into the targeted schema
- Dbt test: runs dbt tests to validate the data and ensure transformations are correct
- Dbt snapshot: capture historical changes in dimension tables
- Error handling: logs or alerts for any task failure



5. Implement dbt models:
    a. Create a dbt project structure - Initialized dbt project names “product_catalog” with standard folders (core, models, snapshots,tests)



    b. Implement a staging model to clean and prepare the data - Implemented a staging model to clean and standardize raw data before transformation. This ensures consistent field names, dropping of null values.

stg_products.sql

Users > simmi > hw3.2_pipeline > dbt_project > product_catalog > models > staging > stg_products.sql

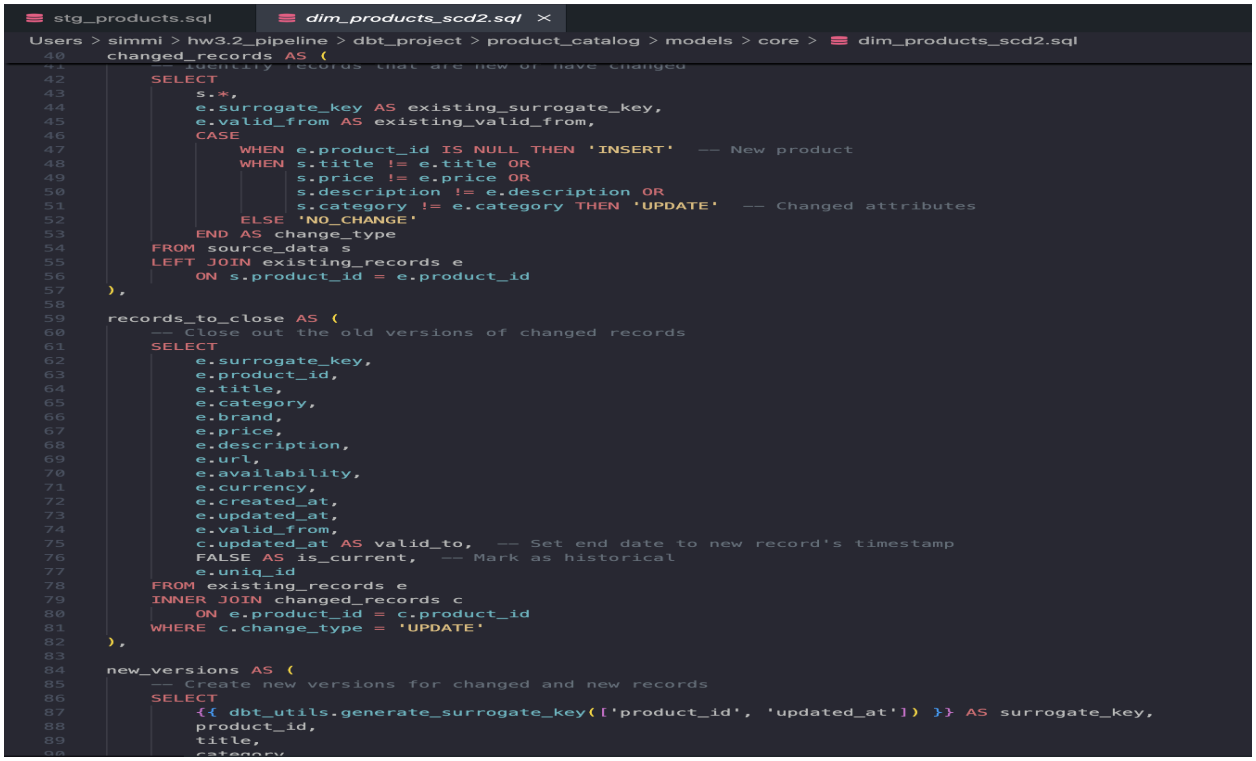
```sql
{{
    config(
        materialized='view',
        schema='staging'
    )
}}

-- Staging model: Clean and prepare raw product data
-- This model standardizes data types, handles nulls, and applies business rules

WITH source_data AS (
    SELECT * FROM {{ source('staging', 'raw_products') }}
),

cleaned_data AS (
    SELECT
        -- Primary identifiers
        product_id,
        uniq_id,
        sku,
        gtin13,

        -- Product information
        TRIM(title) AS title,
        TRIM(COALESCE(NULLIF(brand, ''), 'Unbranded')) AS brand,
        CASE
            WHEN brand IS NULL OR brand = '' THEN 'Unbranded'
            ELSE TRIM(brand)
        END AS category,
        TRIM(description) AS description,
        TRIM(url) AS url,

        -- Pricing and availability
        CAST(price AS DECIMAL(10,2)) AS price,
        UPPER(TRIM(currency)) AS currency,
        TRIM(availability) AS availability,

        -- Timestamps
        created_at,
        updated_at,
        loaded_at

    FROM source_data
    WHERE product_id IS NOT NULL
        AND price IS NOT NULL
        AND price > 0
        AND title IS NOT NULL
)

SELECT * FROM cleaned_data
```

stg_products.sql | schema.yml

Users > simmi > hw3.2_pipeline > dbt_project > product_catalog > models > staging > schema.yml

```yaml
version: 2

sources:
  - name: staging
    description: "Raw staging data from source systems"
    database: PRODUCT_CATALOG
    schema: STAGING
    tables:
      - name: raw_products
        description: "Raw product data loaded from CSV"
        columns:
          - name: product_id
            description: "Unique product identifier"
            tests:
              - not_null
          - name: price
            description: "Product price in local currency"
            tests:
              - not_null
          - name: title
            description: "Product title/name"

models:
  - name: stg_products
    description: "Cleaned and standardized product data from staging"
    columns:
      - name: product_id
        description: "Unique product identifier"
        tests:
          - unique
          - not_null
      - name: title
        description: "Product title"
        tests:
          - not_null
      - name: price
        description: "Product price"
        tests:
          - not_null
      - name: updated_at
        description: "Last update timestamp"
        tests:
          - not_null
```

c. Create a core model that implements SCD Type 2 for the product dimension - developed this to implement SCD Type 2 logic using LAG and LEAD functions to track historical changes.



stg_products.sql | schema.yml

Users > simmi > hw3.2_pipeline > dbt_project > product_catalog > models > core > schema.yml

```yaml
version: 2

models:
  - name: dim_products_scd2
    description: "Product dimension table with SCD Type 2 for tracking historical changes"
    columns:
      - name: surrogate_key
        description: "Surrogate key combining product_id and timestamp"
        tests:
          - unique
          - not_null

      - name: product_id
        description: "Natural business key for the product"
        tests:
          - not_null

      - name: title
        description: "Product title/name"
        tests:
          - not_null

      - name: price
        description: "Product price"
        tests:
          - not_null
          - dbt_utils.accepted_range:
              min_value: 0
              inclusive: false

      - name: valid_from
        description: "Start date when this version became active"
        tests:
          - not_null

      - name: is_current
        description: "Flag indicating if this is the current active record"
        tests:
          - not_null
          - accepted_values:
              values: [true, false]

      - name: category
        description: "Product category"
        tests:
          - not_null

      - name: currency
        description: "Price currency code"
        tests:
```

d. Write appropriate tests for your models - added tests to ensure primary key uniqueness, and correct SCD behaviour.

Users > simmi > hw3.2_pipeline > dbt_project > product_catalog > tests > test_no_overlapping_dates.sql

```sql
-- Test: Ensure no overlapping date ranges for the same product
-- Each product should have non-overlapping valid_from/valid_to ranges

WITH date_ranges AS (
    SELECT
        product_id,
        valid_from,
        COALESCE(valid_to, '9999-12-31'::TIMESTAMP_NTZ) AS valid_to
    FROM {{ ref('dim_products_scd2') }}
)

SELECT
    a.product_id,
    a.valid_from AS range1_start,
    a.valid_to AS range1_end,
    b.valid_from AS range2_start,
    b.valid_to AS range2_end
FROM date_ranges a
JOIN date_ranges b
    ON a.product_id = b.product_id
    AND a.valid_from < b.valid_from
WHERE a.valid_to > b.valid_from
```

Users > simmi > hw3.2_pipeline > dbt_project > product_catalog > tests > test_scd2_integrity.sql

```sql
-- Test: Ensure each product has exactly one current record
-- This test validates SCD Type 2 implementation

SELECT
    product_id,
    COUNT(*) as current_count
FROM {{ ref('dim_products_scd2') }}
WHERE is_current = TRUE
GROUP BY product_id
HAVING COUNT(*) > 1
```

6. Configure dbt snapshots:

Configured dbt snapshots to capture changes in the data over time.

product_snapshot.sql

Users > simmi > hw3.2_pipeline > dbt_project > product_catalog > snapshots > product_snapshot.sql

```sql
{% snapshot product_snapshot %}

{{
    config(
      target_schema='snapshots',
      unique_key='product_id',
      strategy='timestamp',
      updated_at='updated_at',
      invalidate_hard_deletes=True,
    )
}}

-- Snapshot of product data to track historical changes
-- This captures point-in-time product information

SELECT
    product_id,
    title,
    brand,
    category,
    price,
    currency,
    description,
    url,
    availability,
    updated_at,
    uniq_id
FROM {{ ref('stg_products') }}

{% endsnapshot %}
```

7. Integrate dbt with Airflow:

Integrated dbt into Airflow by running dbt run, dbt test, and dbt snapshot via BashOperator tasks in the DAG.

Dbt run -

```
(venv) simmi@Anushkas-MacBook-Air product_catalog % dbt run
08:29:46  Running with dbt=1.8.7
08:29:47  Registered adapter: snowflake=1.8.3
08:29:48  Found 4 models, 24 data tests, 1 snapshot, 1 source, 573 macros
08:29:48
08:29:50  Concurrency: 4 threads (target='dev')
08:29:50
08:29:50  1 of 4 START sql table model STAGING.my_first_dbt_model ....................... [RUN]
08:29:50  2 of 4 START sql view model STAGING_staging.stg_products ...................... [RUN]
08:29:51  2 of 4 OK created sql view model STAGING_staging.stg_products ................. [SUCCESS 1 in 0.91s]
08:29:51  3 of 4 START sql incremental model STAGING_core.dim_products_scd2 ............. [RUN]
08:29:51  1 of 4 OK created sql table model STAGING.my_first_dbt_model .................. [SUCCESS 1 in 1.34s]
08:29:51  4 of 4 START sql view model STAGING.my_second_dbt_model ....................... [RUN]
08:29:53  4 of 4 OK created sql view model STAGING.my_second_dbt_model ................... [SUCCESS 1 in 1.37s]
08:29:55  3 of 4 OK created sql incremental model STAGING_core.dim_products_scd2 ........ [SUCCESS 790 in 4.41s]
08:29:55
08:29:55  Finished running 2 view models, 1 table model, 1 incremental model in 0 hours 0 minutes and 7.52 second
s (7.52s).
08:29:55
08:29:55  Completed successfully
08:29:55
```

Dbt test -

```
(venv) simmi@Anushkas-MacBook-Air product_catalog % dbt test
08:30:14  Running with dbt=1.8.7
08:30:15  Registered adapter: snowflake=1.8.3
08:30:15  Found 4 models, 24 data tests, 1 snapshot, 1 source, 573 macros
08:30:15
08:30:17  Concurrency: 4 threads (target='dev')
08:30:17
08:30:17  1 of 24 START test accepted_values_dim_products_scd2_currency__USD__EUR__GBP__CAD  [RUN]
08:30:17  3 of 24 START test dbt_utils_accepted_range_dim_products_scd2_price__False__0 .. [RUN]
08:30:17  4 of 24 START test not_null_dim_products_scd2_category ......................... [RUN]
08:30:17  2 of 24 START test accepted_values_dim_products_scd2_is_current__True__False ... [RUN]
08:30:18  4 of 24 PASS not_null_dim_products_scd2_category ............................... [PASS in 1.48s]
08:30:18  5 of 24 START test not_null_dim_products_scd2_is_current ....................... [RUN]
08:30:19  1 of 24 PASS accepted_values_dim_products_scd2_currency__USD__EUR__GBP__CAD .... [PASS in 2.31s]
08:30:19  6 of 24 START test not_null_dim_products_scd2_price ............................ [RUN]
08:30:19  2 of 24 PASS accepted_values_dim_products_scd2_is_current__True__False ......... [PASS in 2.46s]
08:30:19  3 of 24 PASS dbt_utils_accepted_range_dim_products_scd2_price__False__0 ........ [PASS in 2.52s]
08:30:19  7 of 24 START test not_null_dim_products_scd2_product_id ....................... [RUN]
08:30:19  5 of 24 PASS not_null_dim_products_scd2_is_current ............................. [PASS in 1.05s]
08:30:19  8 of 24 START test not_null_dim_products_scd2_surrogate_key .................... [RUN]
08:30:19  9 of 24 START test not_null_dim_products_scd2_title ............................ [RUN]
08:30:20  6 of 24 PASS not_null_dim_products_scd2_price .................................. [PASS in 0.95s]
08:30:20  10 of 24 START test not_null_dim_products_scd2_valid_from ....................... [RUN]
08:30:20  7 of 24 PASS not_null_dim_products_scd2_product_id ............................. [PASS in 0.82s]
08:30:20  11 of 24 START test not_null_my_first_dbt_model_id ............................. [RUN]
08:30:20  9 of 24 PASS not_null_dim_products_scd2_title .................................. [PASS in 0.83s]
08:30:20  12 of 24 START test not_null_my_second_dbt_model_id ............................ [RUN]
08:30:21  8 of 24 PASS not_null_dim_products_scd2_surrogate_key .......................... [PASS in 1.29s]
08:30:21  13 of 24 START test not_null_stg_products_price ................................ [RUN]
08:30:21  10 of 24 PASS not_null_dim_products_scd2_valid_from ............................ [PASS in 1.02s]
```

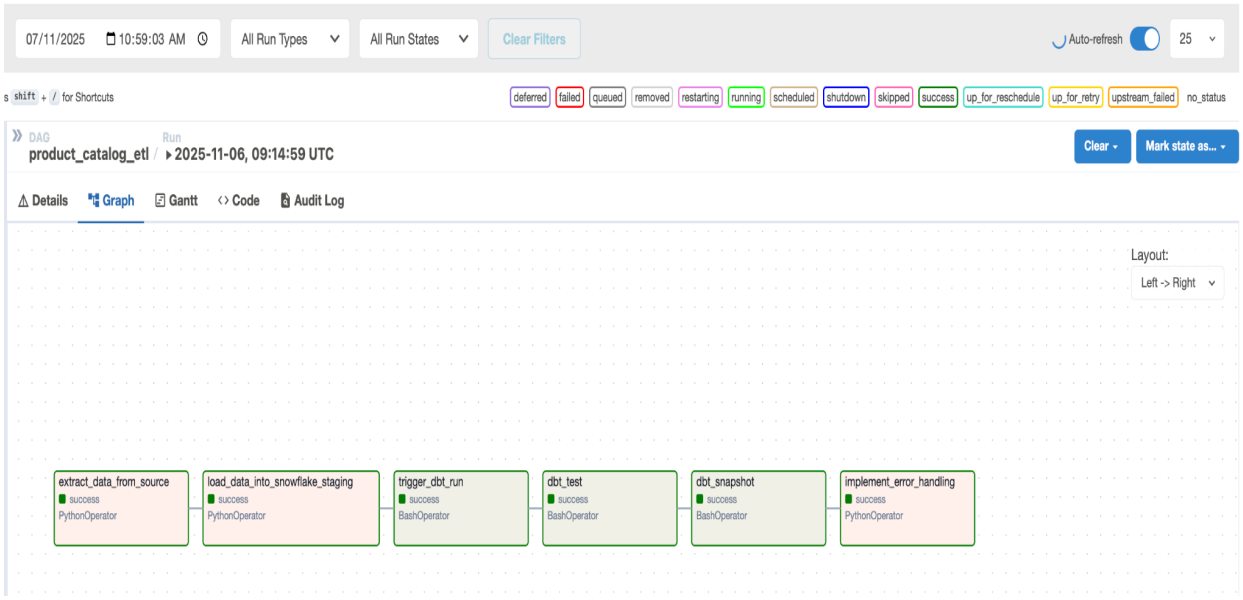
Dbt snapshot -

```
(venv) simmi@Anushkas-MacBook-Air product_catalog % dbt snapshot
08:31:35  Running with dbt=1.8.7
08:31:36  Registered adapter: snowflake=1.8.3
08:31:37  Found 4 models, 24 data tests, 1 snapshot, 1 source, 573 macros
08:31:37
08:31:38  Concurrency: 4 threads (target='dev')
08:31:38
08:31:38  1 of 1 START snapshot snapshots.product_snapshot ............................... [RUN]
08:31:44  1 of 1 OK snapshotted snapshots.product_snapshot ............................... [SUCCESS 636 in 5.52s]
08:31:44
08:31:44  Finished running 1 snapshot in 0 hours 0 minutes and 7.44 seconds (7.44s).
08:31:44
08:31:44  Completed successfully
08:31:44
```

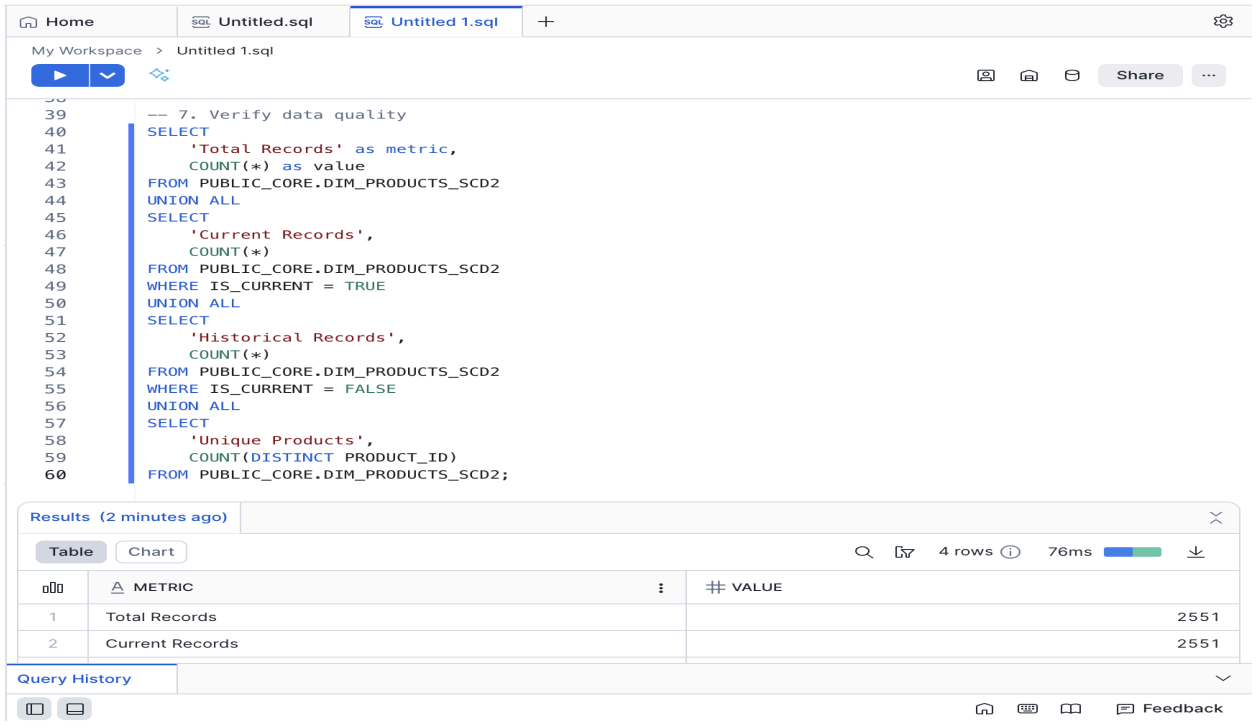
8. Run entire pipeline:

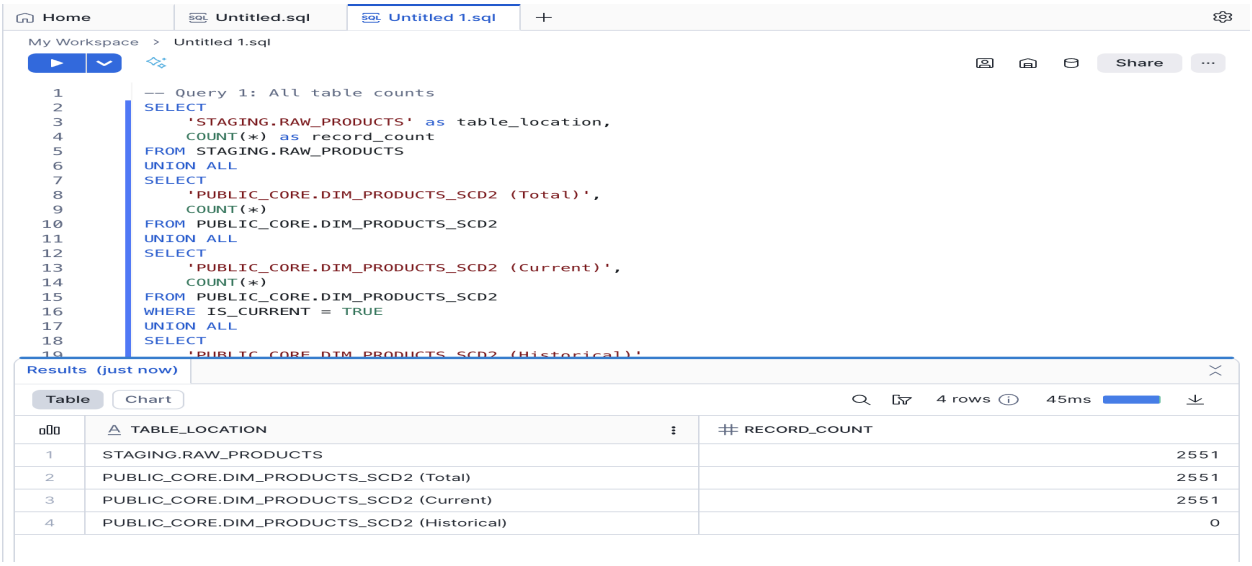
Executed “product_catalog_etl” DAG in the Airflow

a. Run Airflow DAG - all tasks completed successfully

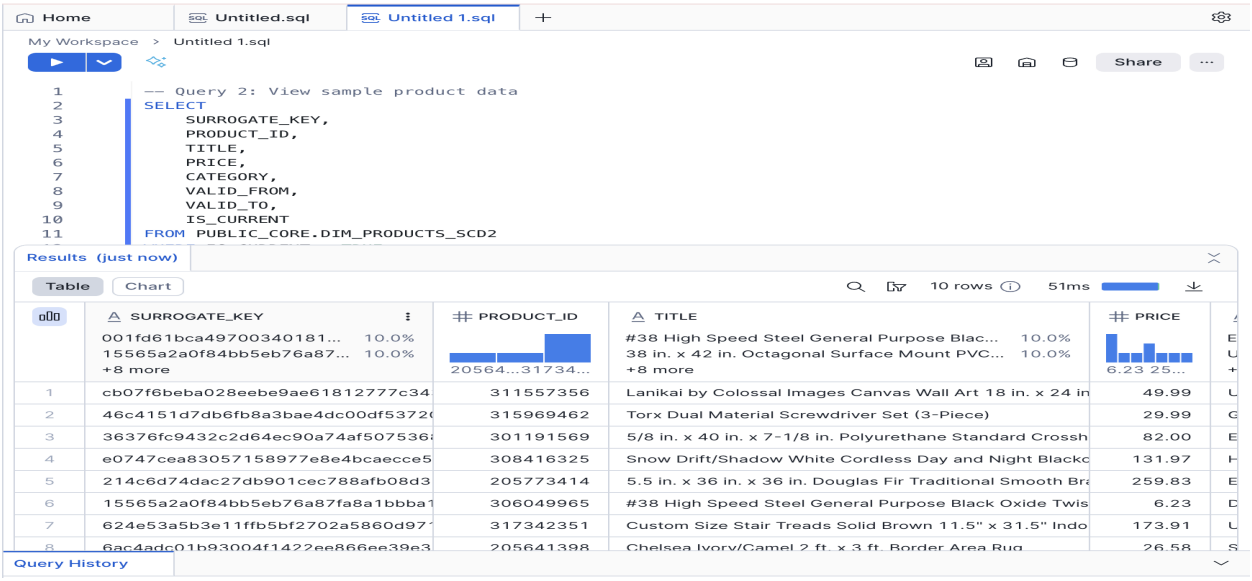


b. Verify data in Snowflake - initialised query in snowflake to confirm transformed product records were loaded correctly.



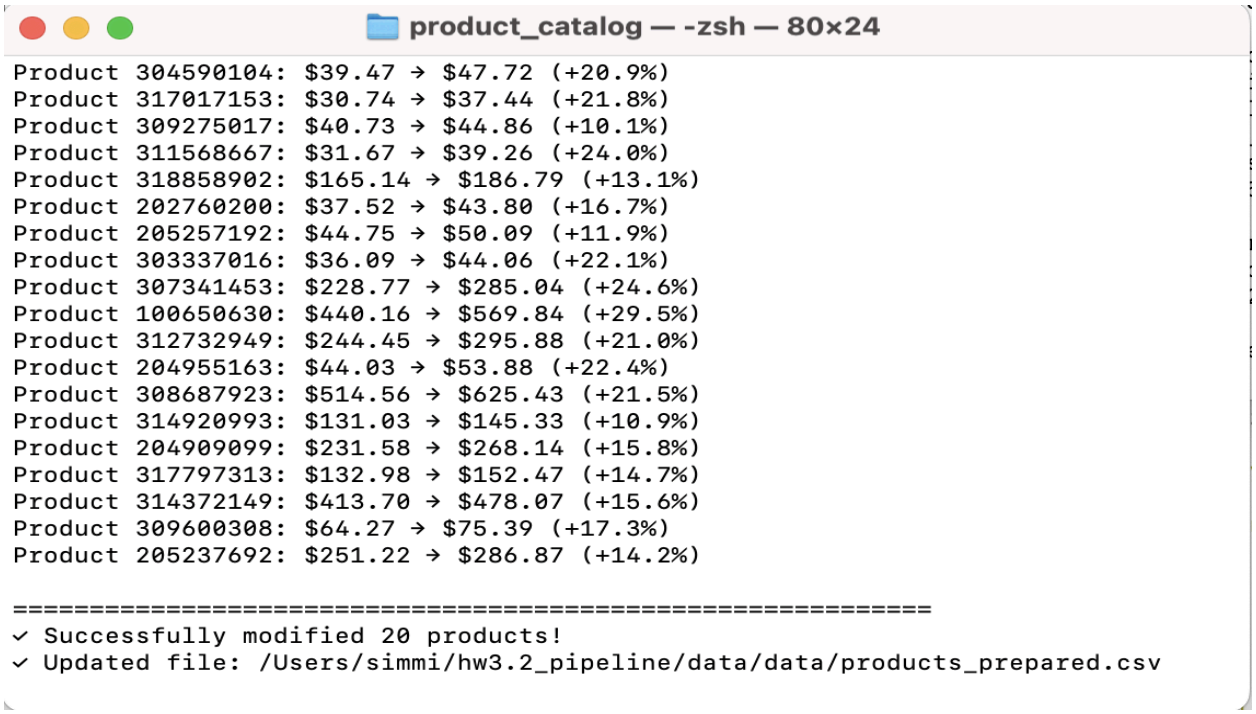


There is zero count for the historical table as we haven't changed our product details yet.

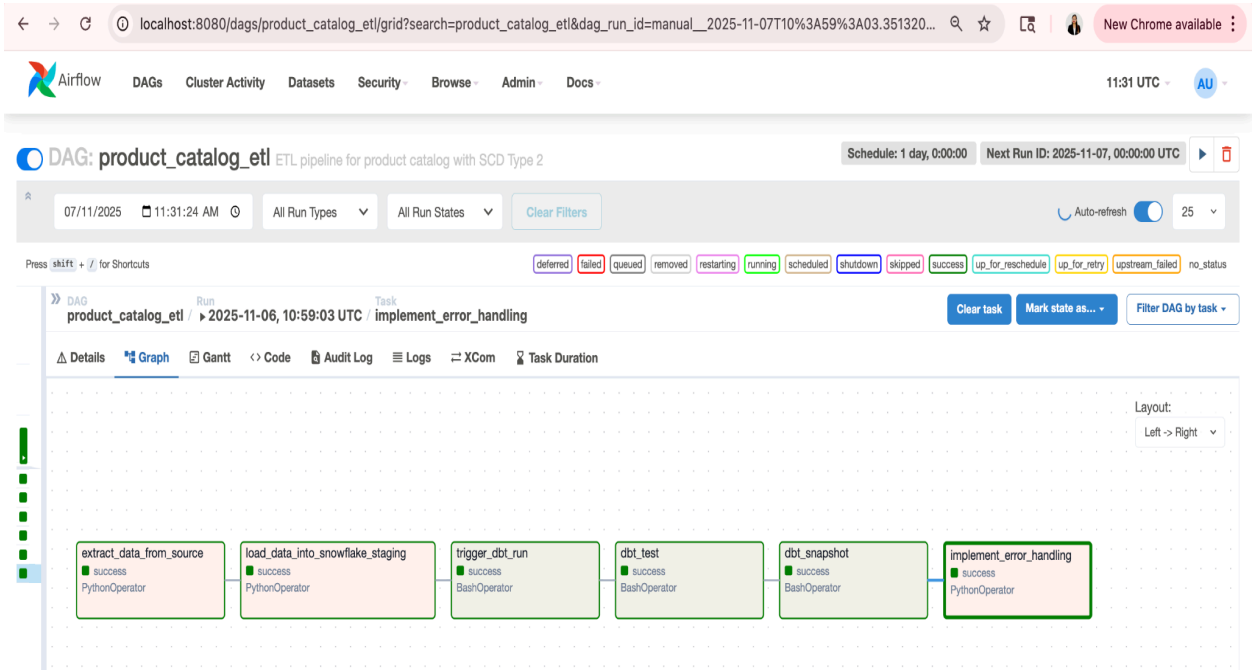


c. Make changes to the source data and re-run the pipeline to test SCD implementation

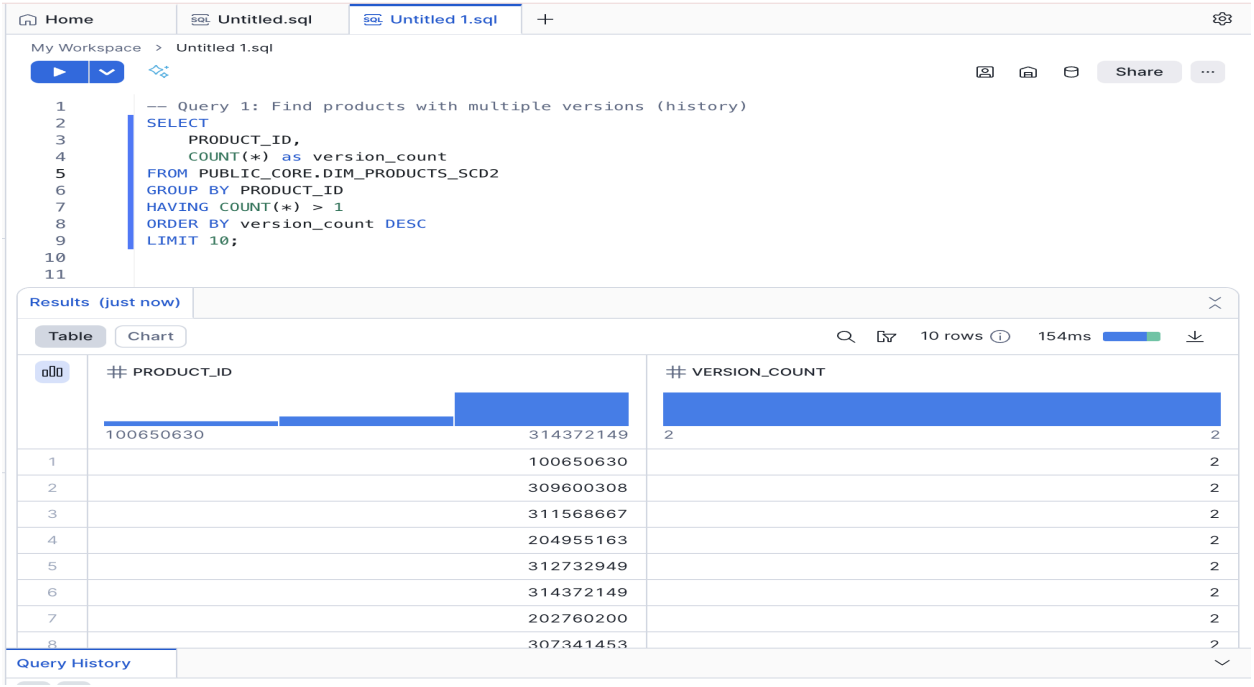
We modified 20 product records and re-ran the pipeline to get historical changes.

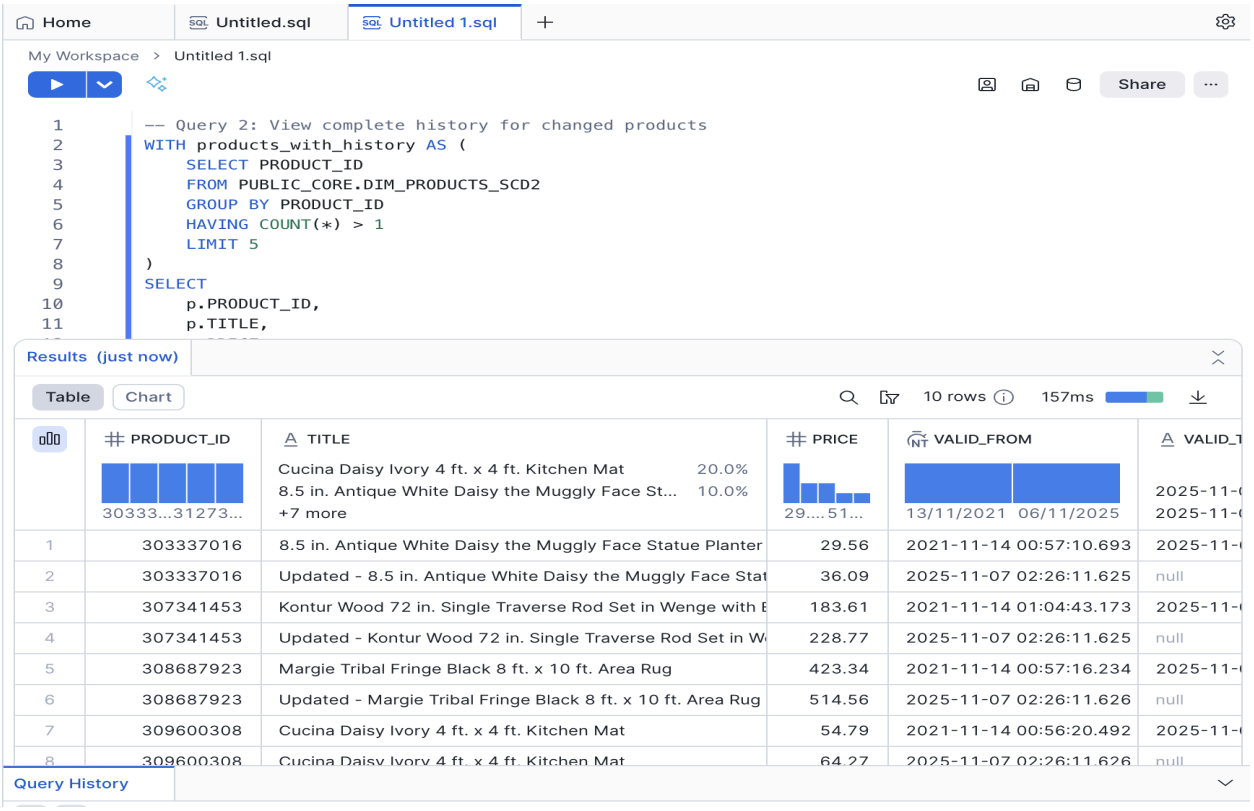


All tasks in the “product_catalog_etl” for SCD Type 2 ran successfully.



Verified the historical entries per product ID using snowflake, validating SCD Type 2 behaviour.





Home | Untitled.sql | Untitled 1.sql

My Workspace > Untitled 1.sql

Share

```sql
-- Query 3: Show price changes over time
WITH ranked_prices AS (
    SELECT
        PRODUCT_ID,
        TITLE,
        PRICE,
        VALID_FROM,
        IS_CURRENT,
        LAG(PRICE) OVER (PARTITION BY PRODUCT_ID ORDER BY VALID_FROM) as previous_price
    FROM PUBLIC_CORE.DIM_PRODUCTS_SCD2
)
```

Results (just now)

Table | Chart

Chart type: Bar chart

X-axis: TITLE

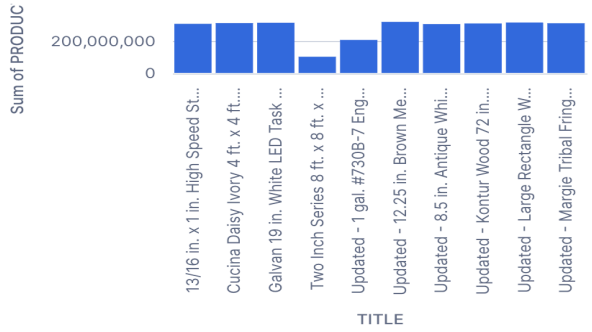
Sort: None

Y-axis: PRODUCT_ID

Add column

Aggregate: Sum

Group by: None



Query History

Home | Untitled.sql | Untitled 1.sql

My Workspace > Untitled 1.sql

Share

```sql
-- Query 4: Audit trail - see all changes
SELECT
    PRODUCT_ID,
    TITLE,
    PRICE,
    CATEGORY,
    VALID_FROM,
    VALID_TO,
    IS_CURRENT,
    DATEDIFF('day', VALID_FROM, COALESCE(VALID_TO, CURRENT_TIMESTAMP())) as days_active
FROM PUBLIC_CORE.DIM_PRODUCTS_SCD2
```

Results (just now)

Table | Chart

6 rows | 109ms

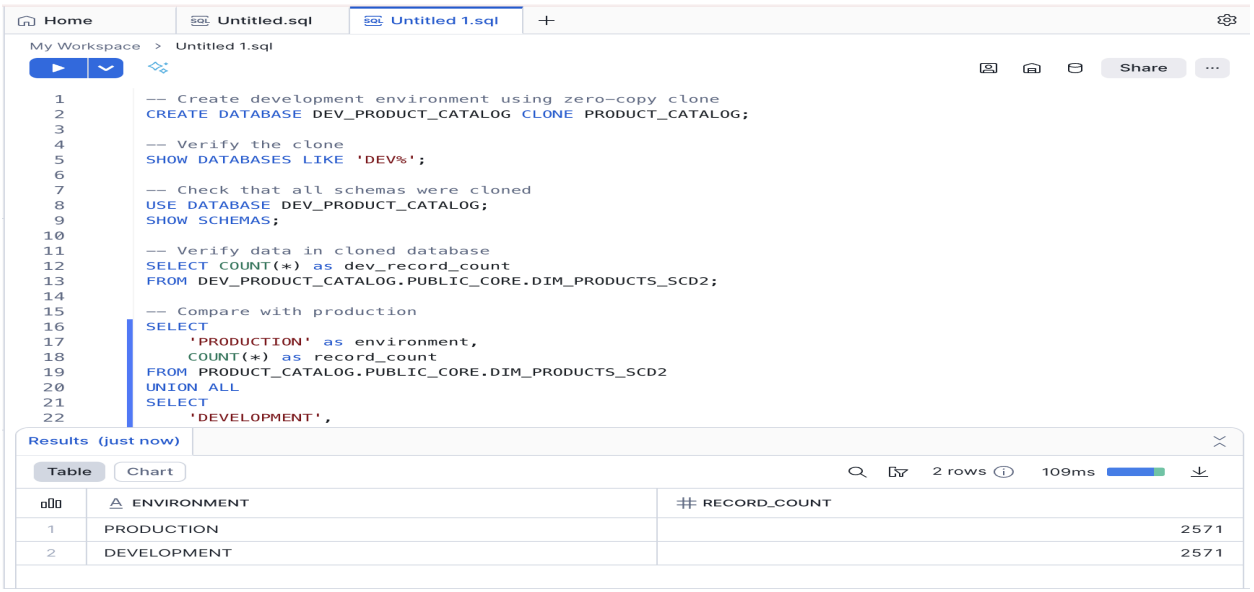
	PRODUCT_ID	TITLE	PRICE	CATEGORY	VALID_FROM
1	205237692	5 gal. #GR-W12 Confident White Semi-Gloss Enamel Exteric	220.00	BEHR ULTRA	2021-11-14 01:04:
2	205237692	5 gal. #GR-W12 Confident White Semi-Gloss Enamel Exteric	251.22	BEHR ULTRA	2025-11-07 02:26:
3	304590104	13/16 in. x 1 in. High Speed Steel Annular Cutter with 3/4 in	32.65	Drill America	2021-11-14 00:56:
4	304590104	13/16 in. x 1 in. High Speed Steel Annular Cutter with 3/4 in	39.47	Drill America	2025-11-07 02:26:
5	314372149	Textile Classic 15 in. 3-Light Brushed Nickel Close-to-Ceilin	358.00	Justice Design	2021-11-14 01:04:
6	314372149	Updated - Textile Classic 15 in. 3-Light Brushed Nickel Clos	413.70	Justice Design	2025-11-07 02:26:

.

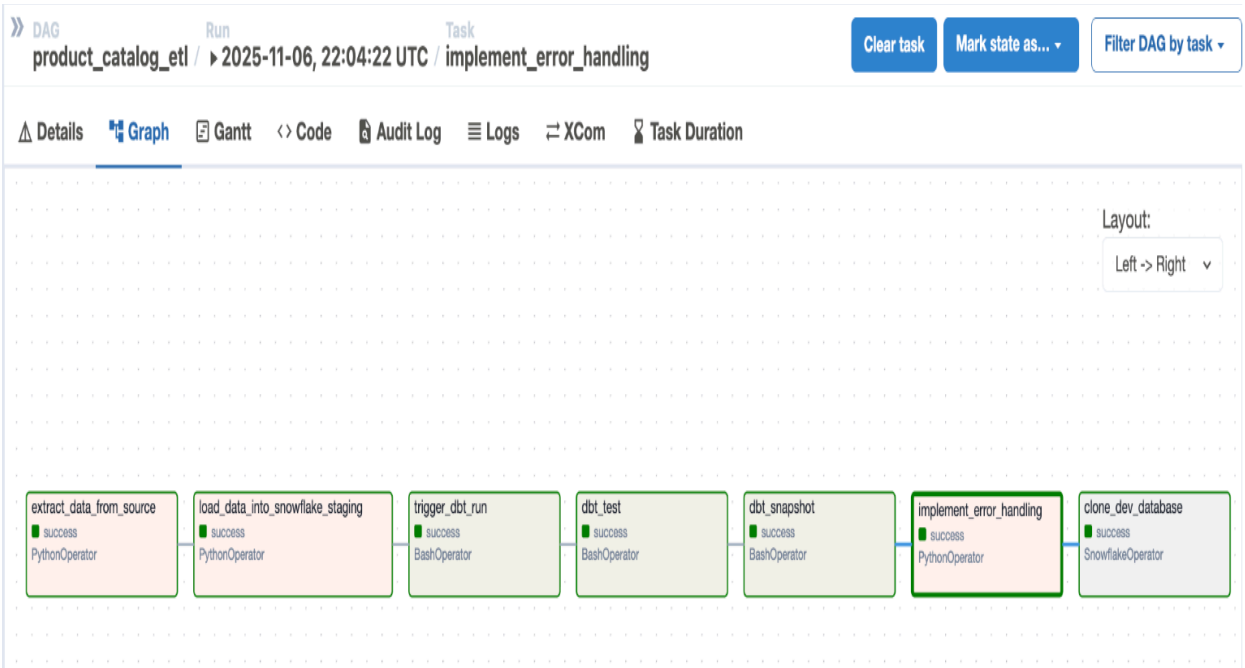
9. Leverage Snowflake Zero-Copy Cloning

a. Development Environment Cloning

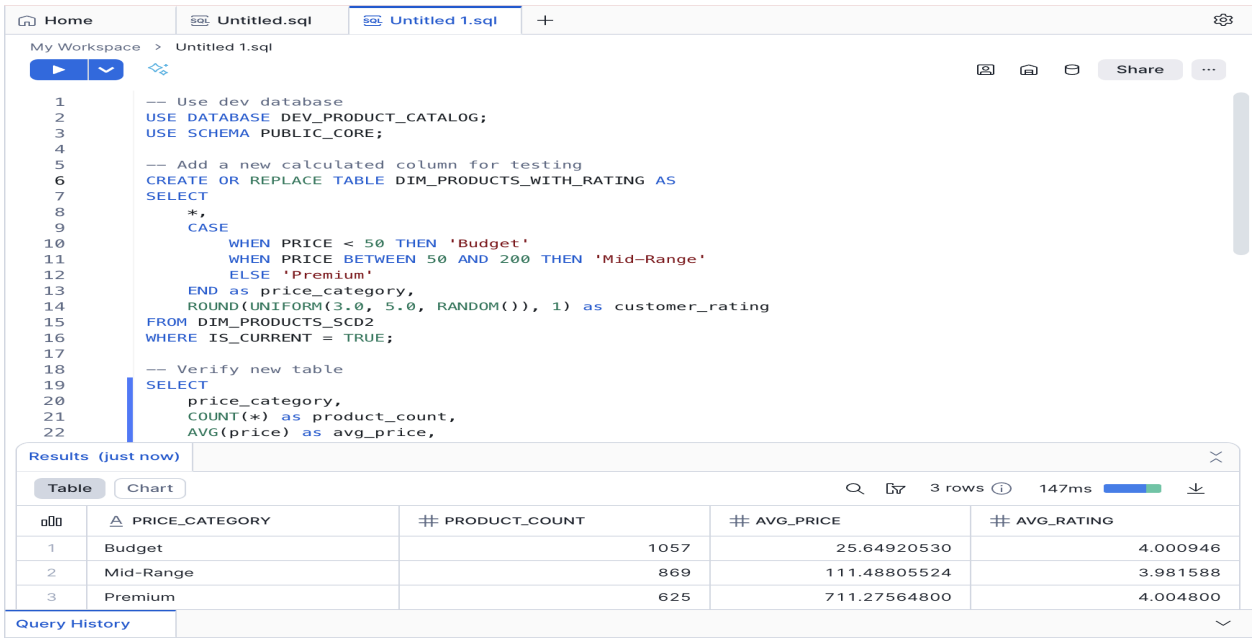
- Create clone database - Used Snowflake's Zero-Copy cloning to create a cloned development database "dev_product_catalog" from production to test changes without affecting original data.



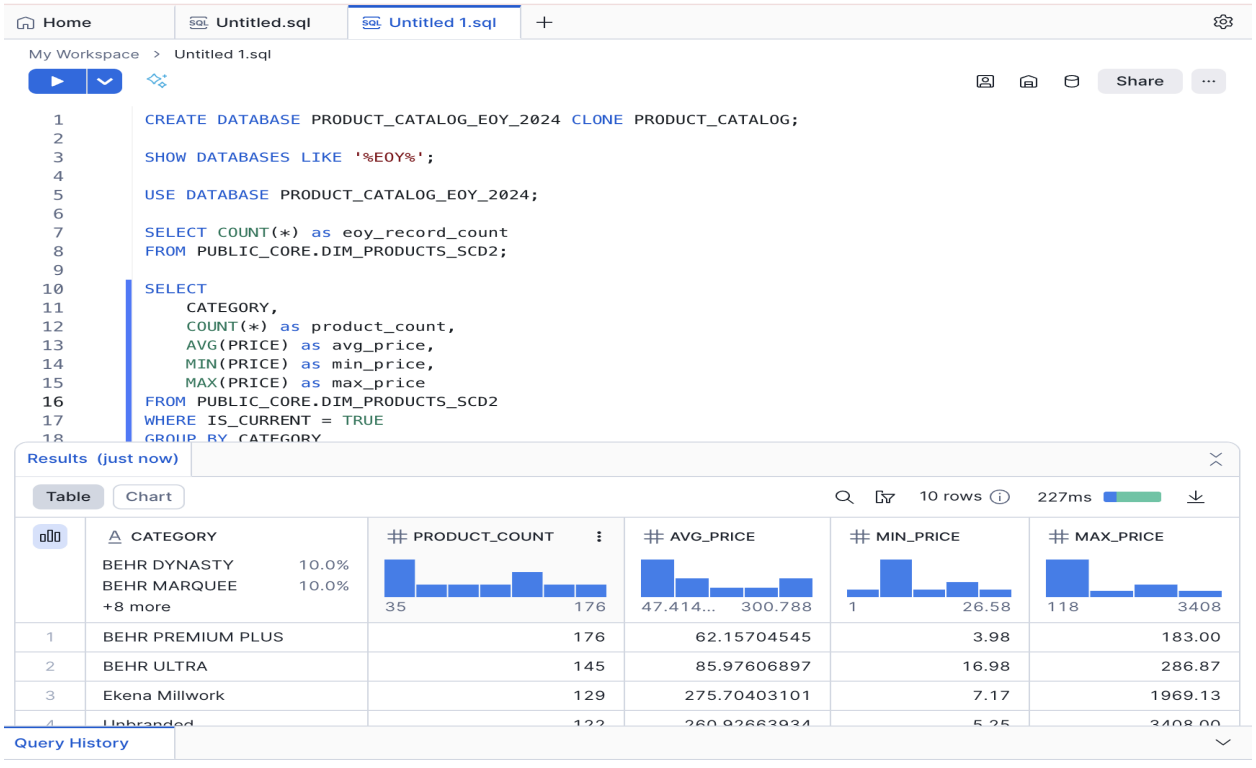
- Modify Airflow DAG that creates this clone



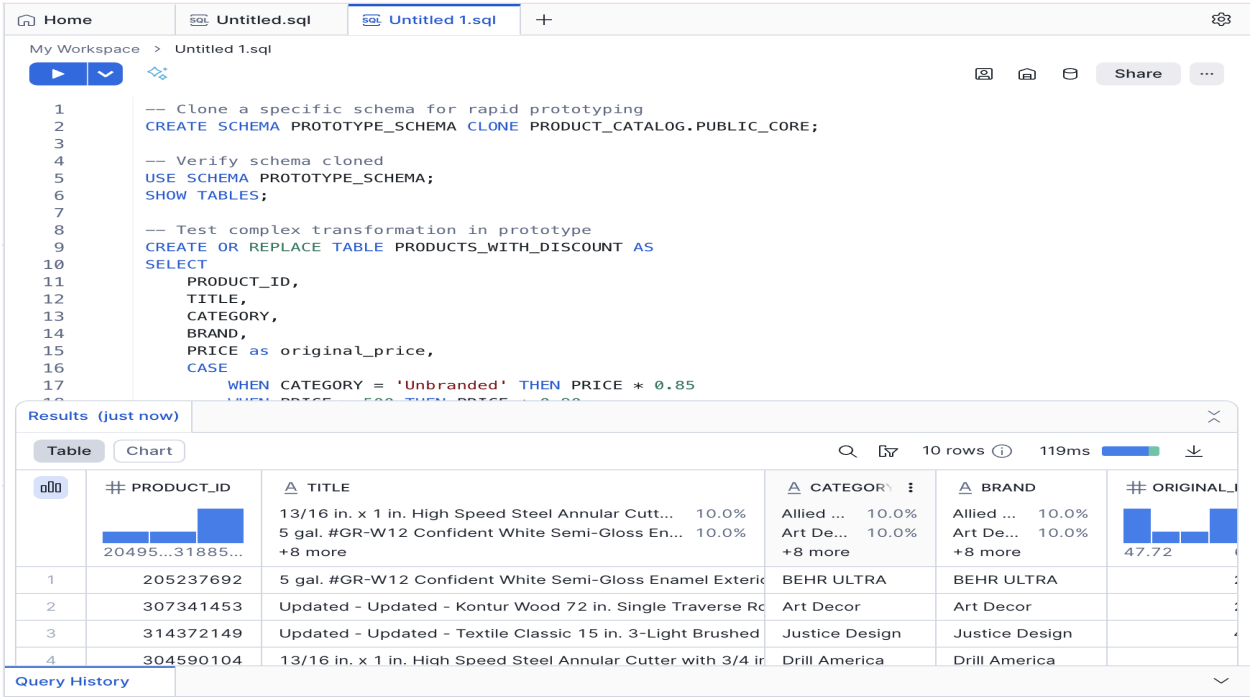
- Implement a dbt model in the development environment to test a new feature-



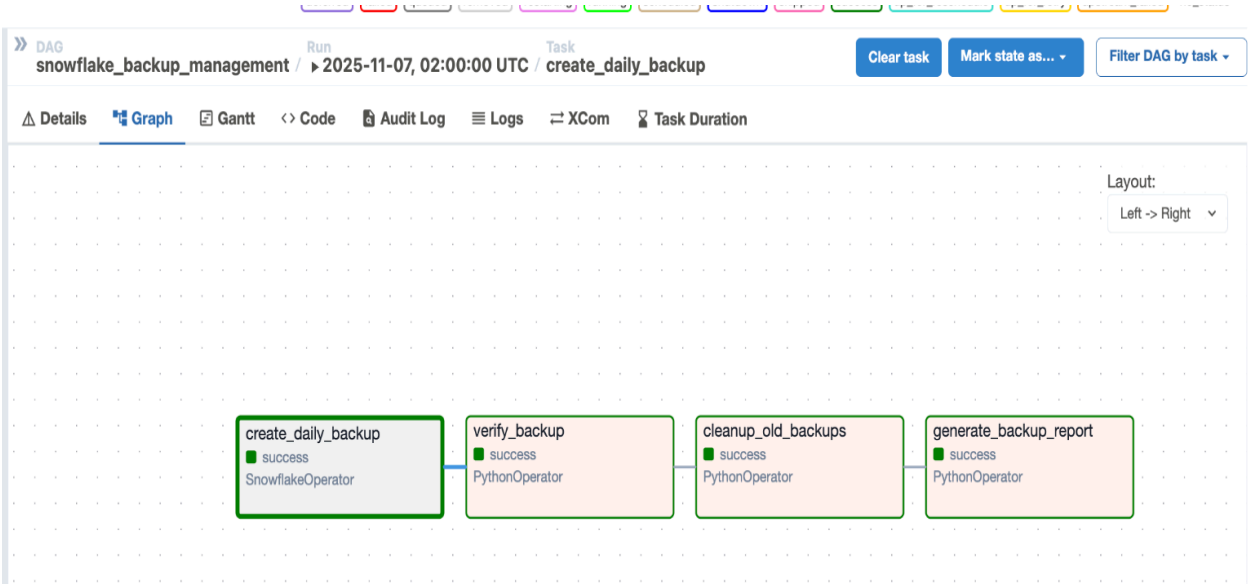
b. Point-in-Time Analysis - Created database clones at specific timestamps to perform historical comparisons and year-end analysis.



c. Rapid Prototyping with Cloning - Cloned a specific schema “prototype_schema” to test new transformations without hindering the existing data.



d. Backup and Restore Strategy - Implemented daily backups using snowflake cloning and configured airflow DAG “snowflake_backup_management”



My Workspace > Untitled 1.sql

Share

```sql
-- Create daily backup using zero-copy clone
CREATE OR REPLACE DATABASE PRODUCT_CATALOG_BACKUP_20250106
CLONE PRODUCT_CATALOG;

-- Verify backup
SHOW DATABASES LIKE '%BACKUP%';

-- Check backup data
SELECT
    'PRODUCTION' as environment,
    COUNT(*) as record_count
FROM PRODUCT_CATALOG.PUBLIC_CORE.DIM_PRODUCTS_SCD2
UNION ALL
SELECT
    'BACKUP',
    COUNT(*)
FROM PRODUCT_CATALOG_BACKUP_20250106.PUBLIC_CORE.DIM_PRODUCTS_SCD2;
```

Results (just now)

Table | Chart — 3 rows — 71ms

	created_on	name	is_default	is_current	origin
1	2025-11-07 13:48:41.590 -080(	PRODUCT_CATALOG_BACKUP_20250104	N	Y	
2	2025-11-07 13:48:37.668 -080(	PRODUCT_CATALOG_BACKUP_20250105	N	N	
3	2025-11-07 13:48:32.618 -080(	PRODUCT_CATALOG_BACKUP_20250106	N	N	

10. Experiment with various options and describe 5 key takeaways.

- Airflow provides robust scheduling and dependency management for ETL orchestration.
- dbt simplifies SQL transformations and enables modular, test-driven data engineering.
- Zero-copy cloning in Snowflake allows for instant environment creation with no additional storage cost.
- SCD Type 2 implementation ensures full product history tracking for analytical accuracy.
- Integrating dbt with Airflow builds an automated end-to-end data pipeline.

11. If given a chance, how will you extend this exercise to make the learning experience better?

1. Deploy the project to a dockerized Airflow environment for portability
2. Integrate CI/CD automation for dbt using GitHub Actions

12. Did you find or come across solutions to similar problems by using Generative AI or other sources?

1. A problem when one Airflow webserver port was occupied and many PID processes were running in the background had been fixed using ChatGPT; GPT-5 instructed me how to find out and kill those using terminal commands.
2. I tried to troubleshoot Airflow's Snowflake connection using ChatGPT (GPT-5) by learning how to correctly configure the snowflake_conn_id in Airflow Connections UI and then how to test it from the command line. This guidance helped resolve the issue where the authentication for SnowflakeOperator failed.