Snowflake Architecture Demo: The "Sony Pictures" Global Data Platform

Target Audience: Sony Pictures Entertainment (SPE) Data Architecture Team
Objective: Demonstrate a modern, governed Medallion Architecture on Snowflake, highlighting the interoperability of imperative (Streams/Tasks) and declarative (Dynamic Tables) paradigms, integrated with dbt and Data Metric Functions (DMFs).

---

1. Demo Architecture Overview (3-10-2 Model)

This demo simulates the processing of global box office data, streaming viewership (Sony Pictures Core), and fan loyalty interactions. We define a precise **3-10-2** table structure.

The Bronze (Raw) Layer: 3 Tables

Landing zone for immutable, raw ingestion.

1. **RAW.SPE.FAN_INTERACTIONS_STREAM**: Real-time stream of fan site logins, "Sony Rewards" signups, and movie reviews (JSON format).
2. **RAW.SPE.TITLE_METADATA_CATALOG**: Master metadata from internal CMS combined with IMDb/Gracenote feeds (Title, Cast, Director, Genre, Runtime) (JSON format).
3. **RAW.SPE.GLOBAL_BOX_OFFICE**: Daily CSV dumps from theatrical distribution partners (AMC, Regal, International Distributors).

The Silver (Refined) Layer: 10 Tables

Cleanse, Conform, and Deduplicate.

- **The Fan Identity Fork (Imperative / Tasks):**
  - *Scenario:* Handling PII and separating "Verified Purchasers" from "General Web Traffic".
  1. SILVER.STG.STG_FAN_VERIFIED: Fans with linked ticket purchases (High Value).
  2. SILVER.STG.STG_FAN_GUEST: Anonymous or unverified interactions.
  3. SILVER.DIMS.DIM_FANS_SCD2: Historical fan dimension (tracking Region/Preference changes over time).
  4. SILVER.SECURE.DIM_FAN_PII_VAULT: Sensitive data (Email, Phone) isolated in a restricted schema.
- **The Content & Performance Chain (Declarative / Dynamic Tables):**
  - *Scenario:* Standardizing Movie Titles and Revenue across regions.
  5. SILVER.STG.STG_TITLES_PARSED: Flattened JSON metadata (Cast/Crew arrays).
  6. SILVER.DIMS.DIM_TITLES_SCD1: "Golden Record" of movie metadata (Type 1).
  7. SILVER.STG.STG_BOX_OFFICE_FLATTEN: Parsed raw sales files.

8. SILVER.STG.STG_BOX_OFFICE_CURRENCY: Normalizing JPY/GBP/EUR to USD.
9. SILVER.STG.STG_BOX_OFFICE_DEDUP: Deduplication logic using QUALIFY.
10. SILVER.FACTS.FACT_DAILY_PERFORMANCE: Final enriched fact table joining Revenue to Titles and Release Dates.

The Gold (Aggregated) Layer: 2 Tables

Business-ready aggregations for Tableau/ThoughtSpot.

1. **GOLD.STUDIO_OPS.AGG_FRANCHISE_PERFORMANCE**: Dynamic Table (Spider-Verse, Ghostbusters, etc.).
2. **GOLD.MARKETING.AGG_FAN_LIFETIME_VALUE**: Dynamic Table (Rolling 365-day spend per fan).

---

2. Deep Dive: The Imperative Path (Streams & Tasks)

Scenario: Processing **Fan Loyalty Data**. Because this involves conditional routing (Splitting high-value Verified Fans from casual traffic) and strict PII compliance, we use **Streams and Tasks**.

Step 2.1: The Split (Conditional Multi-Table Insert)

We use a Task to read from a Stream and "fan out" the data.

SQL

```
-- 1. Create the Stream on the raw JSON landing table
CREATE STREAM raw_fan_stream ON TABLE raw.spe.fan_interactions_stream;

-- 2. Create the Router Task
CREATE OR REPLACE TASK tsk_route_fan_data
  WAREHOUSE = 'transform_wh'
  SCHEDULE = '1 minute'
  WHEN SYSTEM$STREAM_HAS_DATA('raw_fan_stream')
AS
  -- "The Split": Conditional Multi-Table Insert
  INSERT ALL
    -- Path A: Verified Ticket Buyers (Linked to Fandango/Cinema account)
    WHEN raw_json:account_type::STRING = 'VERIFIED_LINKED' THEN
      INTO silver.stg.stg_fan_verified
```

```
    VALUES (fan_id, email, region, 'VERIFIED', event_ts)
  -- Path B: Guest/Anonymous Traffic
  ELSE
    INTO silver.stg.stg_fan_guest
    VALUES (session_id, ip_address, region, 'GUEST', event_ts)
SELECT
  raw_json:id::STRING as fan_id,
  raw_json:session::STRING as session_id,
  raw_json:email::STRING as email,
  raw_json:ip::STRING as ip_address,
  raw_json:region::STRING as region,
  raw_json:event_time::TIMESTAMP as event_ts,
  raw_json
FROM raw_fan_stream;
```

Step 2.2: The Finalizer (SCD Type 2 Merge)

We chain a second task to process the Verified Fans into the Dimension, preserving history (e.g., A fan moves from "UK" to "USA" - we want to track that migration).

SQL

```
CREATE OR REPLACE TASK tsk_merge_fan_scd2
  AFTER tsk_route_fan_data
AS
  MERGE INTO silver.dims.dim_fans_scd2 AS target
  USING silver.stg.stg_fan_verified AS source
  ON target.fan_id = source.fan_id
  -- Complex Logic to close old record (set valid_to) and insert new one
... ;
```

---

3. Deep Dive: The Declarative Path (Dynamic Tables)

Scenario: Processing **Box Office & Title Data**. This is a linear analytical flow (Ingest -> Clean -> Currency Conversion -> Aggregate). We use **Dynamic Tables (DTs)**.

The "Cleanse, Dedup, Lookup" Chain

Step 3.1: Cleanse & Flatten (Table 7)

SQL

```
CREATE DYNAMIC TABLE silver.stg.stg_box_office_flatten
  TARGET_LAG = '5 minutes'
  WAREHOUSE = 'transform_wh'
AS
  SELECT
    raw_data:theater_id::STRING as theater_id,
    raw_data:title_id::STRING as title_id,
    raw_data:local_gross::FLOAT as local_gross_amt,
    raw_data:currency::STRING as currency_code,
    raw_data:show_date::DATE as report_date
  FROM raw.spe.global_box_office;
```

Step 3.2: Currency Normalization & Data Quality (Table 8)

Note: We attach a DMF here later.

SQL

```
CREATE DYNAMIC TABLE silver.stg.stg_box_office_currency
  TARGET_LAG = '5 minutes'
AS
  SELECT
    b.title_id,
    b.report_date,
    -- Simple conversion logic (Real-world would join to a Rate table)
    CASE
      WHEN b.currency_code = 'JPY' THEN b.local_gross_amt * 0.0067
      WHEN b.currency_code = 'GBP' THEN b.local_gross_amt * 1.27
      ELSE b.local_gross_amt
    END AS gross_usd
  FROM silver.stg.stg_box_office_flatten b
  WHERE gross_usd >= 0; -- Basic filtering
```

Step 3.3: Deduplicate (Table 9)

The critical "Dedup" step using window functions.

SQL

```
CREATE DYNAMIC TABLE silver.stg.stg_box_office_dedup
  TARGET_LAG = '5 minutes'
AS
  SELECT *
  FROM silver.stg.stg_box_office_currency
  QUALIFY ROW_NUMBER() OVER (
      PARTITION BY title_id, report_date, theater_id
      ORDER BY ingestion_ts DESC
  ) = 1;
```

- **Why:** QUALIFY ensures that if a theater sends a correction file for the same date, we only process the latest version.[1]

Step 3.4: Dimension Lookup & Fact Creation (Table 10)

Joining the sales stream with the Movie Dimension to create the Star Schema.

SQL

```
CREATE DYNAMIC TABLE silver.facts.fact_daily_performance
  TARGET_LAG = '10 minutes'
AS
  SELECT
    s.report_date,
    s.gross_usd,
    t.title_key,    -- Surrogate Key from Titles Dim
    t.franchise_name -- e.g., 'Spider-Man Universe'
  FROM silver.stg.stg_box_office_dedup s
  LEFT JOIN silver.dims.dim_titles_scd1 t
    ON s.title_id = t.imdb_id;
```

---

4. Governance: Data Metric Functions (DMFs)

We integrate **Data Quality** directly into the pipeline.

Step 1: Define the Metric (Valid IMDb Score)

SQL

```
CREATE DATA METRIC FUNCTION governance.metrics.valid_imdb_score(
  ARG_T FLOAT
)
RETURNS NUMBER AS
$$  SELECT CASE WHEN ARG_T BETWEEN 0 AND 10 THEN 1 ELSE 0 END$$;
```

Step 2: Apply to Title Metadata
We apply this to DIM_TITLES_SCD1 to ensure we don't ingest corrupted rating data.

SQL

```
ALTER TABLE silver.dims.dim_titles_scd1
ADD DATA METRIC FUNCTION governance.metrics.valid_imdb_score(imdb_rating)
ON SCHEDULE '5 MINUTE';
```

Step 3: View Results
Any failures are automatically logged to
SNOWFLAKE.LOCAL.DATA_QUALITY_MONITORING_RESULTS.

---

5. dbt Integration

Why dbt for Sony Pictures?
While Snowflake handles the execution, dbt handles the engineering lifecycle.

1. **Macros for Currency Conversion:** Instead of hardcoding exchange rates in SQL (as seen in Step 3.2), we use a dbt macro {{ convert_currency('amount', 'currency_code') }} ensuring all analysts use the same official Finance logic.

2. **Environment awareness:** Run the pipeline in DEV_SPE_DB during testing and PROD_SPE_DB for release seamlessly.
3. **Docs:** dbt generates a documentation site showing that FACT_DAILY_PERFORMANCE is downstream of DIM_TITLES, helping Business Analysts understand the data lineage.[2]

---

6. Gold Layer Analytics: 10 Example Queries

This section demonstrates the value of the Gold Layer to Sony Pictures executives. We categorize these by complexity.

Simple Queries (The "What Happened?" Layer)

1. Franchise Leaderboard (Box Office)

- *Objective:* Which Franchise is generating the most revenue this quarter?

SQL

```
SELECT
    t.franchise_name,
    SUM(f.gross_usd) as total_global_box_office
FROM gold.studio_ops.agg_franchise_performance f
JOIN silver.dims.dim_titles_scd1 t ON f.title_key = t.title_key
WHERE f.report_date >= DATE_TRUNC('QUARTER', CURRENT_DATE())
GROUP BY 1
ORDER BY 2 DESC;
```

2. Top Rated Action Movies

- *Objective:* Identify high-quality back-catalogue content for Sony Pictures Core promotion.

SQL

```
SELECT
    title_name,
    release_year,
    imdb_rating
```

```
FROM silver.dims.dim_titles_scd1
WHERE genre = 'Action' AND release_year < 2020
ORDER BY imdb_rating DESC
LIMIT 10;
```

3. Fan Regional Distribution

- *Objective:* Where are our "Super Fans" located?

SQL

```
SELECT
  region,
  COUNT(DISTINCT fan_id) as active_verified_fans
FROM silver.dims.dim_fans_scd2
WHERE is_current = TRUE
 AND loyalty_tier = 'PLATINUM'
GROUP BY 1;
```

4. Daily Release Tracking

- *Objective:* How is a specific movie performing day-over-day since release?

SQL

```
SELECT
  report_date,
  gross_usd
FROM silver.facts.fact_daily_performance
WHERE title_key = (SELECT title_key FROM silver.dims.dim_titles_scd1 WHERE title_name =
'Kraven the Hunter')
ORDER BY report_date ASC;
```

Medium Queries (The "Why/Correlation" Layer)

5. ROI Analysis: Marketing vs. Box Office

- *Objective:* Compare marketing spend (ingested from SAP) against opening weekend

actuals.

SQL

```
SELECT
    t.title_name,
    m.marketing_budget,
    SUM(f.gross_usd) as opening_weekend_gross,
    (SUM(f.gross_usd) / m.marketing_budget) * 100 as roi_percentage
FROM silver.facts.fact_daily_performance f
JOIN silver.dims.dim_titles_scd1 t ON f.title_key = t.title_key
JOIN raw.spe.marketing_spend m ON t.title_id = m.title_id
WHERE f.report_date BETWEEN t.release_date AND t.release_date + 2
GROUP BY 1, 2;
```

6. Fan Affinity Analysis (Basket Analysis)

- *Objective:* "If a fan watched *Spider-Man*, what else did they watch?"

SQL

```
WITH spidey_fans AS (
    SELECT DISTINCT fan_key
    FROM silver.facts.fact_watch_events
    WHERE title_key = (SELECT title_key FROM silver.dims.dim_titles_scd1 WHERE title_name =
'Spider-Man: Across the Spider-Verse')
)
SELECT
    t.title_name,
    COUNT(DISTINCT w.fan_key) as shared_viewers
FROM silver.facts.fact_watch_events w
JOIN silver.dims.dim_titles_scd1 t ON w.title_key = t.title_key
WHERE w.fan_key IN (SELECT fan_key FROM spidey_fans)
  AND t.title_name!= 'Spider-Man: Across the Spider-Verse'
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

7. Sentiment Trends for Directors

- *Objective:* Analyze how fan sentiment changes for a director over their last 3 films.

SQL

```sql
SELECT
    t.director,
    t.title_name,
    t.release_year,
    AVG(f.sentiment_score) as avg_fan_sentiment
FROM silver.facts.fact_fan_reviews f
JOIN silver.dims.dim_titles_scd1 t ON f.title_key = t.title_key
WHERE t.director = 'Christopher Nolan' -- Example
GROUP BY 1, 2, 3
ORDER BY 3 ASC;
```

8. SCD2 Analysis: The "Moved Fan" Effect

- *Objective:* Do fans spend more or less after moving regions? (Leveraging SCD2 history).

SQL

```sql
SELECT
    curr.region as new_region,
    prev.region as old_region,
    AVG(curr.annual_spend - prev.annual_spend) as spend_delta
FROM silver.dims.dim_fans_scd2 curr
JOIN silver.dims.dim_fans_scd2 prev
  ON curr.fan_id = prev.fan_id
  AND curr.previous_valid_to = prev.valid_to -- Linking history chains
WHERE curr.region!= prev.region
GROUP BY 1, 2;
```

Complex Queries (The "Predictive/Strategic" Layer)

9. The "Blockbuster Prediction" Model

- *Objective:* Correlate pre-release social sentiment (Bronze Stream) with Post-Release Box Office (Silver Fact) to create a predictive multiplier for future films.

SQL

```
WITH PreReleaseBuzz AS (
    -- Aggregate social sentiment 30 days BEFORE release
    SELECT
        title_id,
        COUNT(*) as mention_volume,
        AVG(sentiment_score) as buzz_score
    FROM raw.spe.fan_interactions_stream
    WHERE event_type = 'SOCIAL_MENTION'
    GROUP BY 1
),
FirstWeekPerformance AS (
    -- Aggregate Revenue 7 days AFTER release
    SELECT
        t.imdb_id as title_id,
        SUM(f.gross_usd) as week1_revenue
    FROM silver.facts.fact_daily_performance f
    JOIN silver.dims.dim_titles_scd1 t ON f.title_key = t.title_key
    WHERE f.report_date <= t.release_date + 7
    GROUP BY 1
)
SELECT
    t.genre,
    CORR(p.buzz_score, w.week1_revenue) as correlation_coefficient,
    -- If buzz is high but revenue is low, we have a marketing misalignment
    AVG(CASE WHEN p.buzz_score > 0.8 THEN w.week1_revenue ELSE NULL END) as
high_buzz_avg_rev
FROM PreReleaseBuzz p
JOIN FirstWeekPerformance w ON p.title_id = w.title_id
JOIN silver.dims.dim_titles_scd1 t ON p.title_id = t.imdb_id
GROUP BY 1;
```

10. "Super Fan" Lifetime Value (LTV) Cohort Analysis

- *Objective:* Calculate LTV across Box Office, Streaming, and Merch, grouped by the year they joined the loyalty program.

SQL

```
SELECT
    DATE_TRUNC('YEAR', d.join_date) as cohort_year,
    COUNT(DISTINCT d.fan_key) as cohort_size,
    -- Aggregate spend from multiple fact tables
    SUM(COALESCE(bo.ticket_spend, 0) + COALESCE(str.sub_spend, 0) +
COALESCE(mer.merch_spend, 0)) as total_cohort_revenue,
    (total_cohort_revenue / cohort_size) as avg_ltv
FROM silver.dims.dim_fans_scd2 d
-- Left Join to Box Office Spend
LEFT JOIN (
    SELECT fan_key, SUM(amount) as ticket_spend FROM silver.facts.fact_ticket_sales GROUP BY 1
) bo ON d.fan_key = bo.fan_key
-- Left Join to Streaming Spend
LEFT JOIN (
    SELECT fan_key, SUM(amount) as sub_spend FROM silver.facts.fact_streaming_subs GROUP
BY 1
) str ON d.fan_key = str.fan_key
-- Left Join to Merchandise Spend
LEFT JOIN (
    SELECT fan_key, SUM(amount) as merch_spend FROM silver.facts.fact_merch_sales GROUP BY
1
) mer ON d.fan_key = mer.fan_key
WHERE d.is_current = TRUE
GROUP BY 1
ORDER BY 1 DESC;
```

---

7. Summary: Decision Matrix for SPE

Feature	Dynamic Tables (Declarative)	Streams & Tasks (Imperative)
Best Used For	Continuous pipelines (Global Box Office $\to$ Silver $\to$ Gold). "I want	Complex orchestration, routing, or non-SQL logic. "Check if Fan is Verified; if

	the daily revenue numbers, keep them fresh."	yes, PII Vault; if no, Guest Table."
Data Logic	**Linear Chaining:** Great for currency conversion, parsing Cast/Crew JSON arrays, and aggregating franchise totals.	**Splitting/Routing:** INSERT ALL allows splitting the Fan Stream into multiple security zones (Public vs. PII).
SCD Strategy	**SCD Type 1:** Auto-updates Movie Metadata (e.g., Runtime correction).	**SCD Type 2:** Tracking Fan migration (Region changes) for historical compliance.
Maintenance	**Low:** Snowflake manages the scheduler.	**Medium:** You manage the DAG dependencies.
Our Demo	Used for **Content & Revenue** (High volume, linear transformation).	Used for **Fan Loyalty & PII** (Complex validation, routing, History tracking).

Works cited

1. Deduplicating CDC records in Snowflake: The fastest way | by Emiliano Mancuso - Medium, accessed January 21, 2026, https://medium.com/@emancu/deduplicating-cdc-records-in-snowflake-the-fastest-way-5a4a14f9890a
2. Accelerating Data Teams with dbt Cloud & Snowflake, accessed January 21, 2026, https://www.snowflake.com/en/developers/guides/data-teams-with-dbt-cloud/