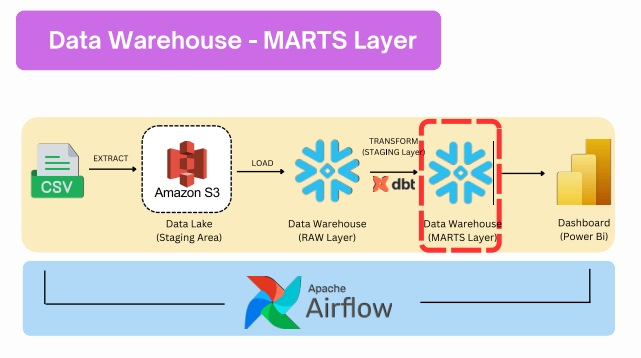
Phase 10: Dimensional Modeling (Star Schema)

Created	@February 13, 2026 7:02 PM
Tags	

จุดประสงค์ใน Phase 10

สร้าง Dimensional Models ในรูปแบบ Star Schema เพื่อให้ Dashboard และ Analytics สามารถ Query ข้อมูลได้อย่างมีประสิทธิภาพ



Project Structure

```
movies_dbt/
├── models/
│   ├── staging/
│   │   ├── sources.yml
│   │   ├── schema.yml
│   │   ├── stg_movies_base.sql
│   │   └── stg_movies.sql
│   │   └── data_quality_report.sql
│   │
│   └── marts/
│       ├── dimensions/                ← Phase 10
│       │   ├── dim_movies.sql
│       │   ├── dim_genres.sql
│       │   ├── dim_directors.sql
│       │   ├── dim_actors.sql
│       │   ├── dim_countries.sql
│       │   ├── dim_languages.sql
│       │   └── dim_time.sql
│       │
│       ├── bridges/                   ← Phase 11
│       │   ├── bridge_movie_genre.sql
│       │   ├── bridge_movie_actor.sql
│       │   ├── bridge_movie_country.sql
│       │   ├── bridge_movie_language.sql
│       │   └── bridge_movie_director.sql
│       │
│       ├── facts/                    ← Phase 11
│       │   └── fact_movie_performance.sql
│       │
│       └── schema.yml             ← Tests & docs (Phase 12)
```

Dimension Tables ที่จะสร้างทั้งหมด (7 tables):

```
1. dim_movies
2. dim_genres
3. dim_directors
4. dim_actors
5. dim_countries
6. dim_languages
7. dim_time
```

วิธีสร้าง DIMENSION TABLES

STEP 1: Setup Folders

1. **Windows CMD:**

```
cd D:\movies_pipeline\movies_dbt

# สร้างโฟลเดอร์
mkdir models\marts
mkdir models\marts\dimensions
mkdir models\marts\bridges
mkdir models\marts\facts
```

2. **Verify:**

```
dir models\marts
# ควรเห็น: dimensions, bridges, facts
```

STEP 2: dim_movies (Main Dimension)

1. **สร้างไฟล์:**

```
code models\marts\dimensions\dim_movies.sql
```

2. **Code:**

```
{{
    config(
        materialized='table',
        schema='marts'
    )
}}

-- ============================================================================
-- MODEL: dim_movies
-- PURPOSE: Movie dimension with all attributes
-- INPUT: {{ ref('stg_movies_enriched') }}
-- OUTPUT: Complete movie information ready for fact tables
-- ============================================================================

with movies as (
    select * from {{ ref('stg_movies_enriched') }}
),
```

```
final as (
    select
        -- ============================================================================
        -- PRIMARY KEY
        -- ============================================================================
        movie_id,

        -- ============================================================================
        -- MOVIE ATTRIBUTES
        -- ============================================================================
        movie_title,
        release_year,
        decade,
        era,

        -- ============================================================================
        -- RATINGS
        -- ============================================================================
        imdb_rating,
        rating_category,
        rotten_tomatoes_pct,
        metacritic_score,

        -- ============================================================================
        -- MOVIE DETAILS
        -- ============================================================================
        runtime_mins,
        runtime_category,
        oscars_won,
        oscar_category,
        box_office_millions,
        box_office_category,

        -- ============================================================================
        -- SINGLE-VALUE TEXT FIELDS (from enriched)
        -- Note: These have 'Unknown' for NULL values
        -- ============================================================================
        director as primary_director,
        -- Note: This is the COALESCE(director_raw, 'Unknown') from enriched
        -- For multiple directors, will need to go back to stg_movies_cleaned

        country as primary_country,
        -- Note: This is the COALESCE(country_raw, 'Unknown') from enriched

        language as primary_language,
        -- Note: This is the COALESCE(language_raw, 'Unknown') from enriched

        -- ============================================================================
        -- MULTI-VALUE FIELDS (Raw - for bridge tables in Phase 11)
        -- These will be used for splitting into bridge tables
        -- ============================================================================
        genres_raw,
        -- Example: "Action|Crime|Drama"

        actors_raw,
        -- Example: "Christian Bale|Heath Ledger"

        country_list,
```

```
        -- Note: This is country_raw from stg_movies_cleaned
        -- Example: "United States|United Kingdom"

        language_list,
        -- Note: This is language_raw from stg_movies_cleaned
        -- Example: "English|German|Polish"

        -- ============================================================================
        -- AUDIT FIELDS
        -- ============================================================================
        loaded_at,
        dbt_updated_at,
        current_timestamp() as dim_created_at

    from movies
)

select * from final
order by movie_id

-- ============================================================================
-- EXPECTED OUTPUT:
-- - 95 rows (all movies from stg_movies_enriched)
-- - Ready for fact table joins
-- - Contains both single values (primary_*) and raw multi-values (*_list, *_raw)
-- ============================================================================
```

3. **Run:**

```
dbt run --select dim_movies
```

4. **Verify:**

```
USE DATABASE movies_db;
USE SCHEMA analytics_marts;

-- ============================================================================
-- dim_movies
-- ============================================================================
SELECT COUNT(*) FROM dim_movies;
-- Expected: 95 rows

SELECT * FROM dim_movies LIMIT 5;
```

ผลลัพธ์ที่ได้:

MOVIE_ID	MOVIE_TITLE	RELEASE_YEAR	DECADE	ERA	IMDB_RATING	RATING_CATEGORY
1	The Shawshank Redemption	1994	1990	1990s	9.3	Masterpiece
2	The Godfather	1972	1970	1970s	9.2	Masterpiece
3	The Dark Knight	2008	2000	2000s	9	Masterpiece
4	The Godfather: Part II	1974	1970	1970s	9	Masterpiece

5	12 Angry Men	1957	1950	1950s	9	Masterpiece

STEP 3: dim_genres

1. สร้างไฟล์:

```
code models\marts\dimensions\dim_genres.sql
```

2. **Code:**

```
{{
    config(
        materialized='table',
        schema='marts'
    )
}}

-- ============================================================================
-- MODEL: dim_genres
-- PURPOSE: Genre lookup dimension
-- INPUT: {{ ref('stg_movies_enriched') }}
-- OUTPUT: Unique list of genres with surrogate keys
-- ============================================================================

with movies as (
    select * from {{ ref('stg_movies_enriched') }}
),

-- STEP 1: Split genres_raw into individual genres
split_genres as (
    select
        movie_id,
        trim(genre.value) as genre_name
    from movies,
    lateral flatten(split(genres_raw, '|')) as genre
    -- LATERAL FLATTEN: Split "Action|Crime|Drama" → 3 rows
    -- SPLIT: Split by "|"
    -- FLATTEN: Convert array → rows
    where genres_raw is not null
),

-- STEP 2: Get unique genres
unique_genres as (
    select distinct
        genre_name
    from split_genres
    where genre_name is not null
      and trim(genre_name) != ''
),

-- STEP 3: Add surrogate key
final as (
    select
        row_number() over (order by genre_name) as genre_id,
        -- Surrogate key: 1, 2, 3, ...
```

```
        genre_name,
        -- Genre name: Action, Drama, Sci-Fi, etc.

        current_timestamp() as dim_created_at

    from unique_genres
)

select * from final
order by genre_name

-- ============================================================================
-- EXPECTED OUTPUT:
-- - ~20 rows (unique genres)
-- - genre_id: 1, 2, 3, ...
-- - genre_name: Action, Adventure, Animation, ...
-- ============================================================================
```

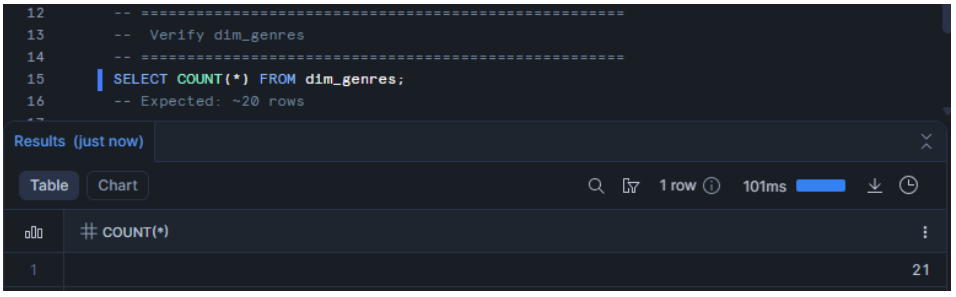
3. **Run:**

```
dbt run --select dim_genres
```

4. **Verify:**

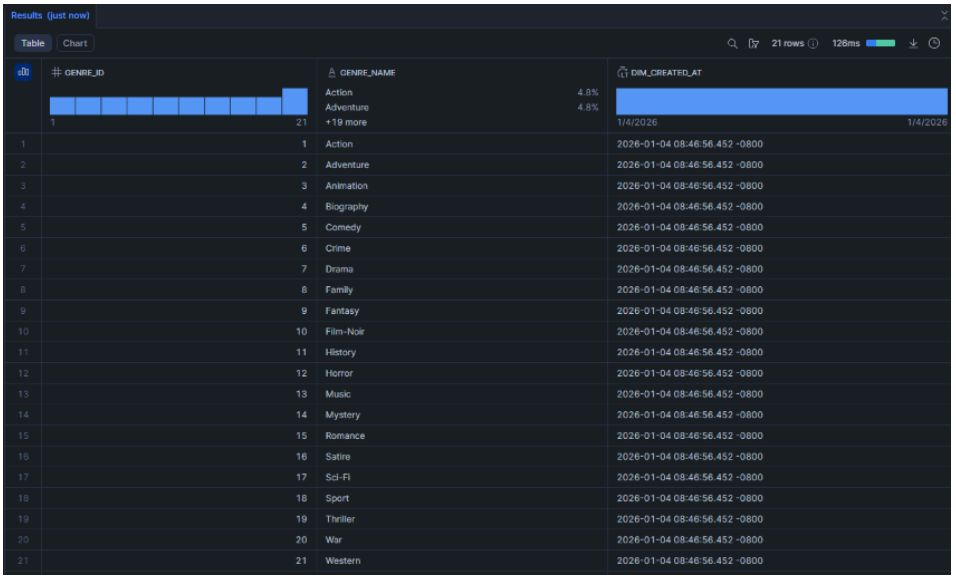
```
-- ============================================================================
-- dim_genres
-- ============================================================================
SELECT COUNT(*) FROM dim_genres;
```

ผลลัพธ์ที่ได้:




```
SELECT * FROM dim_genres ORDER BY genre_name;
-- Should see: Action, Adventure, Animation, Biography, Crime, ..
```

ผลลัพธ์ที่ได้:



STEP 4: dim_directors

1. **สร้างไฟล์:**

```
code models\marts\dimensions\dim_directors.sql
```

2. Code:

```
{{
    config(
        materialized='table',
        schema='marts'
    )
}}

-- ============================================================================
-- MODEL: dim_directors
-- PURPOSE: Director lookup dimension
-- INPUT: {{ ref('stg_movies_cleaned') }}  -- ใช้ cleaned เพราะต้องใช้ director_raw
-- OUTPUT: Unique list of directors with surrogate keys
-- ============================================================================

with movies as (
    select * from {{ ref('stg_movies_cleaned') }}
),

-- STEP 1: Split directors (some movies have multiple directors)
split_directors as (
    select
        movie_id,
        trim(director.value) as director_name
    from movies,
    lateral flatten(split(director_raw, '|')) as director
    -- Split directors separated by "|"
    -- Example: "Joel Coen|Ethan Coen" → 2 rows
    where director_raw is not null
),

-- STEP 2: Get unique directors
```

```
unique_directors as (
    select distinct
        director_name
    from split_directors
    where director_name is not null
      and trim(director_name) != ''
),

-- STEP 3: Add surrogate key
final as (
    select
        row_number() over (order by director_name) as director_id,
        director_name,
        current_timestamp() as dim_created_at
    from unique_directors
)

select * from final
order by director_name

-- ================================================================================
-- EXPECTED OUTPUT:
-- - ~80-90 rows (unique directors)
-- - director_id: 1, 2, 3, ...
-- - director_name: Alfred Hitchcock, Christopher Nolan, ...
-- ================================================================================
```

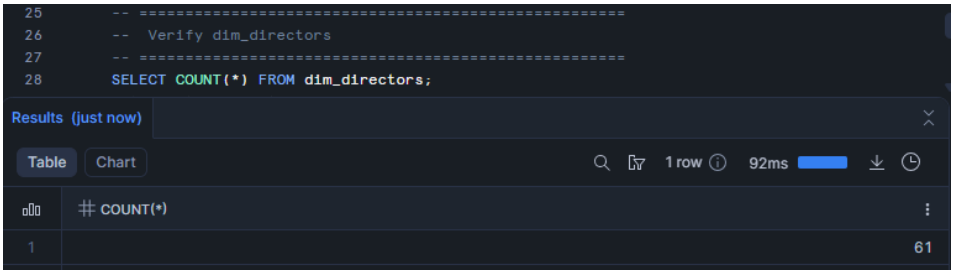
3. **Run:**

```
dbt run --select dim_directors
```

4. **Verify:**

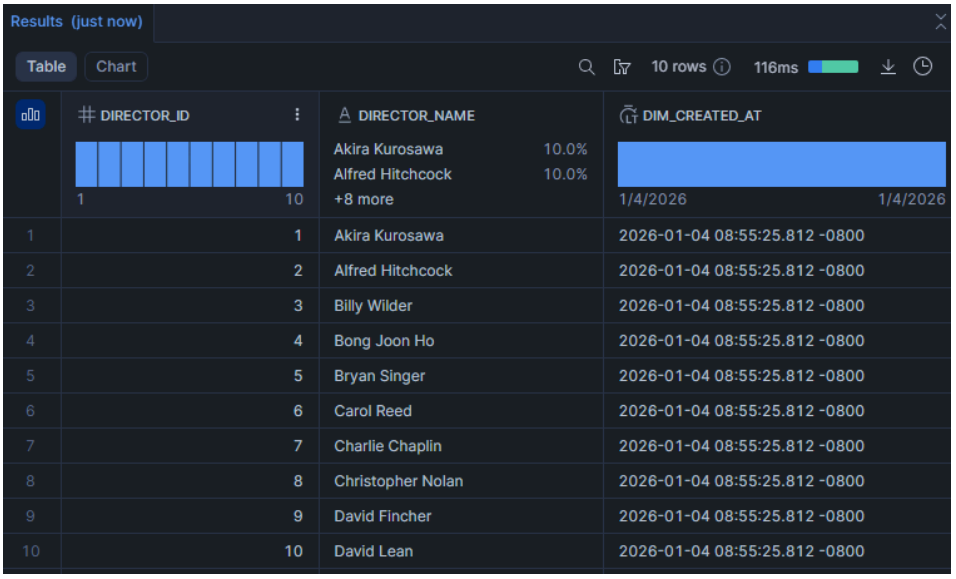
```
-- ================================================================================
-- dim_directors
-- ================================================================================
SELECT COUNT(*) FROM dim_directors;
```

ผลลัพธ์ที่ได้:




```
SELECT * FROM dim_directors LIMIT 10;
```

ผลลัพธ์ที่ได้:



	DIRECTOR_ID	DIRECTOR_NAME	DIM_CREATED_AT
1	1	Akira Kurosawa	2026-01-04 08:55:25.812 -0800
2	2	Alfred Hitchcock	2026-01-04 08:55:25.812 -0800
3	3	Billy Wilder	2026-01-04 08:55:25.812 -0800
4	4	Bong Joon Ho	2026-01-04 08:55:25.812 -0800
5	5	Bryan Singer	2026-01-04 08:55:25.812 -0800
6	6	Carol Reed	2026-01-04 08:55:25.812 -0800
7	7	Charlie Chaplin	2026-01-04 08:55:25.812 -0800
8	8	Christopher Nolan	2026-01-04 08:55:25.812 -0800
9	9	David Fincher	2026-01-04 08:55:25.812 -0800
10	10	David Lean	2026-01-04 08:55:25.812 -0800

STEP 5: dim_actors

1. **สร้างไฟล์:**

```
code models\marts\dimensions\dim_actors.sql
```

2. **Code:**

```
{{
    config(
        materialized='table',
        schema='marts'
    )
}}

-- ============================================================================
-- MODEL: dim_actors
-- PURPOSE: Actor lookup dimension
-- INPUT: {{ ref('stg_movies_enriched') }}
-- OUTPUT: Unique list of actors with surrogate keys
-- ============================================================================

with movies as (
    select * from {{ ref('stg_movies_enriched') }}
),

-- STEP 1: Split actors (pipe-separated)
split_actors as (
    select
        movie_id,
        trim(actor.value) as actor_name
    from movies,
    lateral flatten(split(actors_raw, '|')) as actor
    -- Split actors separated by "|"
    -- Example: "Christian Bale|Heath Ledger" → 2 rows
    where actors_raw is not null
),

-- STEP 2: Get unique actors
```

```
unique_actors as (
    select distinct
        actor_name
    from split_actors
    where actor_name is not null
      and trim(actor_name) != ''
),

-- STEP 3: Add surrogate key
final as (
    select
        row_number() over (order by actor_name) as actor_id,
        actor_name,
        current_timestamp() as dim_created_at
    from unique_actors
)

select * from final
order by actor_name

-- ============================================================================
-- EXPECTED OUTPUT:
-- - ~150-200 rows (unique actors)
-- - actor_id: 1, 2, 3, ...
-- - actor_name: Al Pacino, Brad Pitt, ...
-- ============================================================================
```

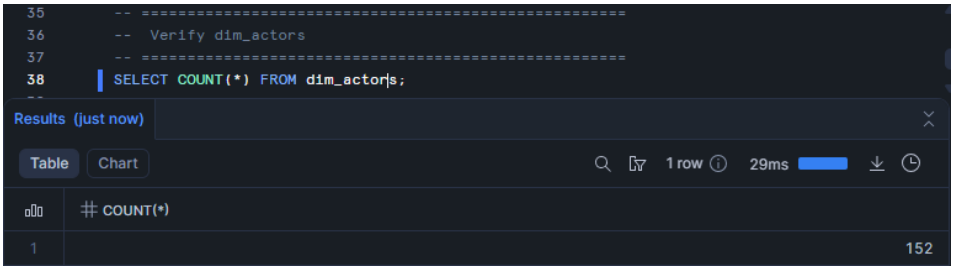
3. **Run:**

```
dbt run --select dim_actors
```

4. **Verify:**

```
-- ============================================================================
-- dim_actors
-- ============================================================================
SELECT COUNT(*) FROM dim_actors;
```

ผลลัพธ์ที่ได้:




```
SELECT * FROM dim_actors LIMIT 20;
```

ผลลัพธ์ที่ได้:

	ACTOR_ID	ACTOR_NAME	DIM_CREATED_AT
1	1	Adolphe Menjou	2026-01-04 08:59:34.877 -0800
2	2	Adrien Brody	2026-01-04 08:59:34.877 -0800
3	3	Al Pacino	2026-01-04 08:59:34.877 -0800
4	4	Alan Rickman	2026-01-04 08:59:34.877 -0800
5	5	Alec Guinness	2026-01-04 08:59:34.877 -0800
6	6	Alexandre Rodrigues	2026-01-04 08:59:34.877 -0800
7	7	Andrew Garfield	2026-01-04 08:59:34.877 -0800
8	8	Anne Bancroft	2026-01-04 08:59:34.877 -0800
9	9	Anne Hathaway	2026-01-04 08:59:34.877 -0800
10	10	Anthony Hopkins	2026-01-04 08:59:34.877 -0800
11	11	Anthony Michael Hall	2026-01-04 08:59:34.877 -0800
12	12	Arnold Schwarzenegger	2026-01-04 08:59:34.877 -0800
13	13	Audrey Tautou	2026-01-04 08:59:34.877 -0800
14	14	Barbara Stanwyck	2026-01-04 08:59:34.877 -0800
15	15	Bengt Ekerot	2026-01-04 08:59:34.877 -0800
16	16	Bibi Andersson	2026-01-04 08:59:34.877 -0800
17	17	Brad Pitt	2026-01-04 08:59:34.877 -0800
18	18	Bruce Willis	2026-01-04 08:59:34.877 -0800
19	19	Carrie-Anne Moss	2026-01-04 08:59:34.877 -0800
20	20	Cary Elwes	2026-01-04 08:59:34.877 -0800

STEP 6: dim_countries

1. **สร้างไฟล์:**

```
code models\marts\dimensions\dim_countries.sql
```

2. **Code:**

```
{{
    config(
        materialized='table',
        schema='marts'
    )
}}

-- ============================================================================
-- MODEL: dim_countries
-- PURPOSE: Country lookup dimension
-- INPUT: {{ ref('stg_movies_enriched') }}
-- OUTPUT: Unique list of countries with surrogate keys
-- ============================================================================

with movies as (
    select * from {{ ref('stg_movies_enriched') }}
),

-- STEP 1: Split countries (pipe-separated)
split_countries as (
    select
        movie_id,
        trim(country.value) as country_name
    from movies,
    lateral flatten(split(country_list, '|')) as country
    -- Note: stg_movies_enriched has "country_list" (not country_raw)
    -- Split "United States|United Kingdom" → 2 rows
    where country_list is not null
),
```

```
-- STEP 2: Get unique countries
unique_countries as (
    select distinct
        country_name
    from split_countries
    where country_name is not null
      and trim(country_name) != ''
),

-- STEP 3: Add surrogate key
final as (
    select
        row_number() over (order by country_name) as country_id,
        country_name,
        current_timestamp() as dim_created_at
    from unique_countries
)

select * from final
order by country_name

-- ================================================================================
-- EXPECTED OUTPUT:
-- - ~15-20 rows (unique countries)
-- - country_id: 1, 2, 3, ...
-- - country_name: Brazil, France, Italy, Japan, United States, ...
-- ================================================================================
```

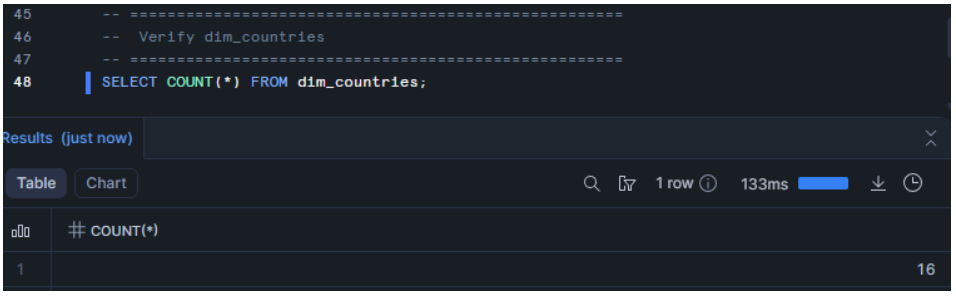
3. **Run:**

```
dbt run --select dim_countries
```

4. **Verify:**

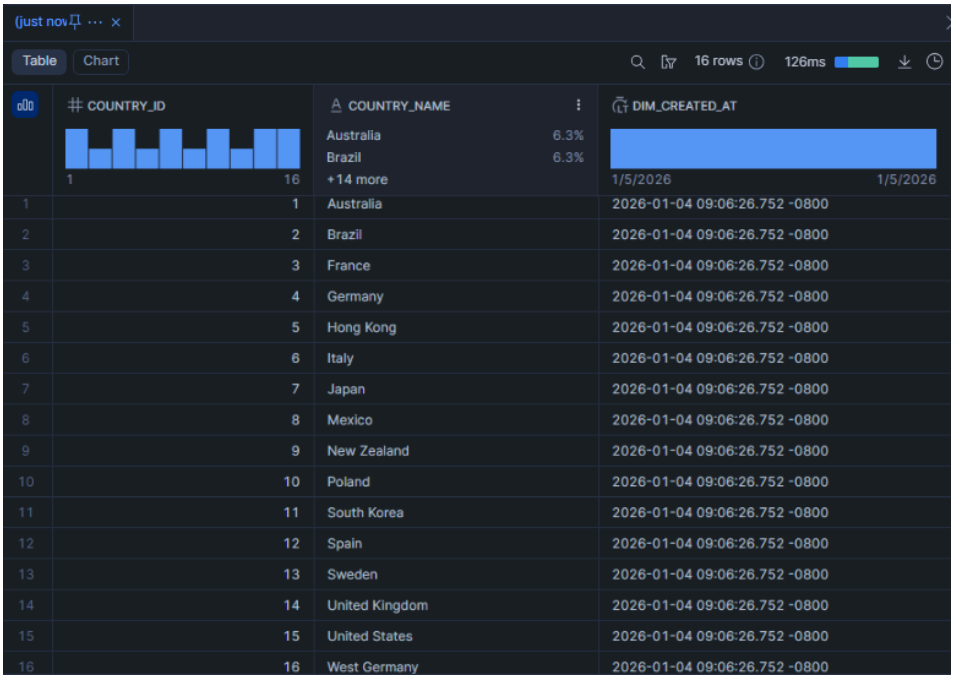
```
-- ================================================================================
-- dim_countries
-- ================================================================================
SELECT COUNT(*) FROM dim_countries;
```

ผลลัพธ์ที่ได้:




```
SELECT * FROM dim_countries ORDER BY country_name;
```

ผลลัพธ์ที่ได้:



	COUNTRY_ID	COUNTRY_NAME	DIM_CREATED_AT
	1 – 16	Australia 6.3%, Brazil 6.3%, +14 more	1/5/2026 – 1/5/2026
1	1	Australia	2026-01-04 09:06:26.752 -0800
2	2	Brazil	2026-01-04 09:06:26.752 -0800
3	3	France	2026-01-04 09:06:26.752 -0800
4	4	Germany	2026-01-04 09:06:26.752 -0800
5	5	Hong Kong	2026-01-04 09:06:26.752 -0800
6	6	Italy	2026-01-04 09:06:26.752 -0800
7	7	Japan	2026-01-04 09:06:26.752 -0800
8	8	Mexico	2026-01-04 09:06:26.752 -0800
9	9	New Zealand	2026-01-04 09:06:26.752 -0800
10	10	Poland	2026-01-04 09:06:26.752 -0800
11	11	South Korea	2026-01-04 09:06:26.752 -0800
12	12	Spain	2026-01-04 09:06:26.752 -0800
13	13	Sweden	2026-01-04 09:06:26.752 -0800
14	14	United Kingdom	2026-01-04 09:06:26.752 -0800
15	15	United States	2026-01-04 09:06:26.752 -0800
16	16	West Germany	2026-01-04 09:06:26.752 -0800

STEP 7: dim_languages

1. **สร้างไฟล์:**

```
code models\marts\dimensions\dim_languages.sql
```

2. **Code:**

```
{{
    config(
        materialized='table',
        schema='marts'
    )
}}

-- ============================================================================
-- MODEL: dim_languages
-- PURPOSE: Language lookup dimension
-- INPUT: {{ ref('stg_movies_enriched') }}
-- OUTPUT: Unique list of languages with surrogate keys
-- ============================================================================

with movies as (
    select * from {{ ref('stg_movies_enriched') }}
),

-- STEP 1: Split languages (pipe-separated)
split_languages as (
    select
        movie_id,
        trim(language.value) as language_name
    from movies,
    lateral flatten(split(language_list, '|')) as language
    -- Note: stg_movies_enriched has "language_list" (not language_raw)
    -- Split "English|German|Polish" → 3 rows
    where language_list is not null
```

```
),

-- STEP 2: Get unique languages
unique_languages as (
    select distinct
        language_name
    from split_languages
    where language_name is not null
      and trim(language_name) != ''
),

-- STEP 3: Add surrogate key
final as (
    select
        row_number() over (order by language_name) as language_id,
        language_name,
        current_timestamp() as dim_created_at
    from unique_languages
)

select * from final
order by language_name

-- ================================================================================
-- EXPECTED OUTPUT:
-- - ~10-15 rows (unique languages)
-- - language_id: 1, 2, 3, ...
-- - language_name: English, French, German, Italian, Japanese, ...
-- ================================================================================
```

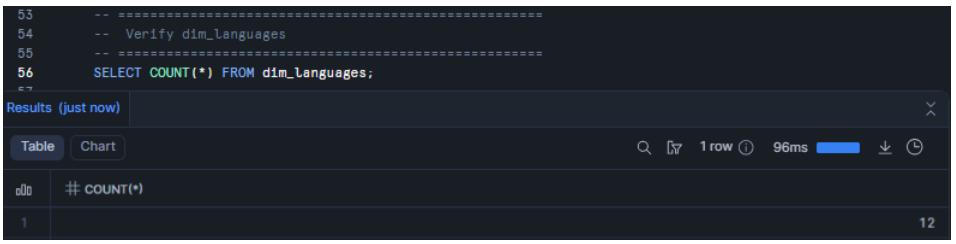
3. **Run:**

```
dbt run --select dim_languages
```

4. **Verify:**

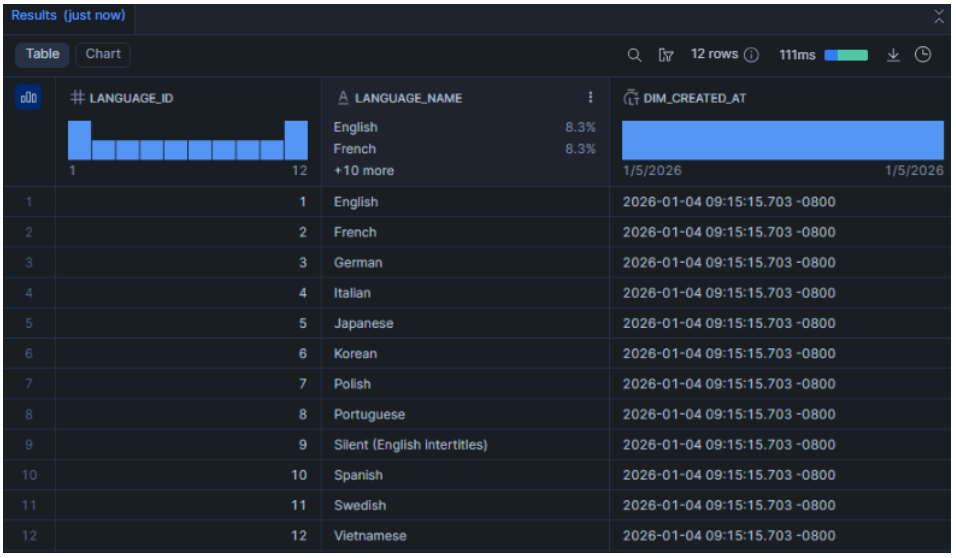
```
-- ================================================================================
-- dim_languages
-- ================================================================================
SELECT COUNT(*) FROM dim_languages;
```

ผลลัพธ์ที่ได้:




```
SELECT * FROM dim_languages ORDER BY language_name;
```

ผลลัพธ์ที่ได้:



	LANGUAGE_ID	LANGUAGE_NAME	DIM_CREATED_AT
1	1	English	2026-01-04 09:15:15.703 -0800
2	2	French	2026-01-04 09:15:15.703 -0800
3	3	German	2026-01-04 09:15:15.703 -0800
4	4	Italian	2026-01-04 09:15:15.703 -0800
5	5	Japanese	2026-01-04 09:15:15.703 -0800
6	6	Korean	2026-01-04 09:15:15.703 -0800
7	7	Polish	2026-01-04 09:15:15.703 -0800
8	8	Portuguese	2026-01-04 09:15:15.703 -0800
9	9	Silent (English intertitles)	2026-01-04 09:15:15.703 -0800
10	10	Spanish	2026-01-04 09:15:15.703 -0800
11	11	Swedish	2026-01-04 09:15:15.703 -0800
12	12	Vietnamese	2026-01-04 09:15:15.703 -0800

STEP 8: dim_time

1. **สร้างไฟล์:**

```
code models\marts\dimensions\dim_time.sql
```

2. **Code:**

```
{{
    config(
        materialized='table',
        schema='marts'
    )
}}

-- ============================================================================
-- MODEL: dim_time
-- PURPOSE: Time dimension (year-based for movies)
-- INPUT: {{ ref('stg_movies_enriched') }}
-- OUTPUT: Year-level time dimension
-- ============================================================================

with movies as (
    select * from {{ ref('stg_movies_enriched') }}
),

-- Get unique years
unique_years as (
    select distinct
        release_year as year
    from movies
    where release_year is not null
),

-- Add time attributes
final as (
    select
        year as time_id,
        -- PK: Year itself (1940, 1950, ...)
```

```
        year,
        -- Full year: 1994, 2010

        floor(year / 10) * 10 as decade,
        -- Decade: 1990, 2000, 2010

        case
            when year < 1960 then 'Classic Era'
            when year < 1980 then 'New Hollywood Era'
            when year < 2000 then 'Blockbuster Era'
            else 'Modern Era'
        end as era,
        -- Movie era

        case
            when year < 2000 then '20th Century'
            else '21st Century'
        end as century,
        -- Century

        -- Additional time attributes
        case
            when year < 1950 then 'Pre-1950s'
            when year < 1960 then '1950s'
            when year < 1970 then '1960s'
            when year < 1980 then '1970s'
            when year < 1990 then '1980s'
            when year < 2000 then '1990s'
            when year < 2010 then '2000s'
            else '2010s+'
        end as decade_label,

        current_timestamp() as dim_created_at

    from unique_years
)

select * from final
order by year

-- ============================================================================
-- EXPECTED OUTPUT:
-- - ~70 rows (one per unique year: 1931-2019)
-- - time_id = year
-- - decade, era, century attributes
-- ============================================================================
```

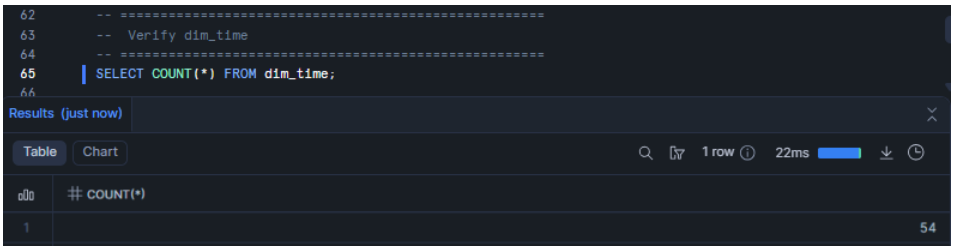
3. **Run:**

```
dbt run --select dim_time
```

4. **Verify:**

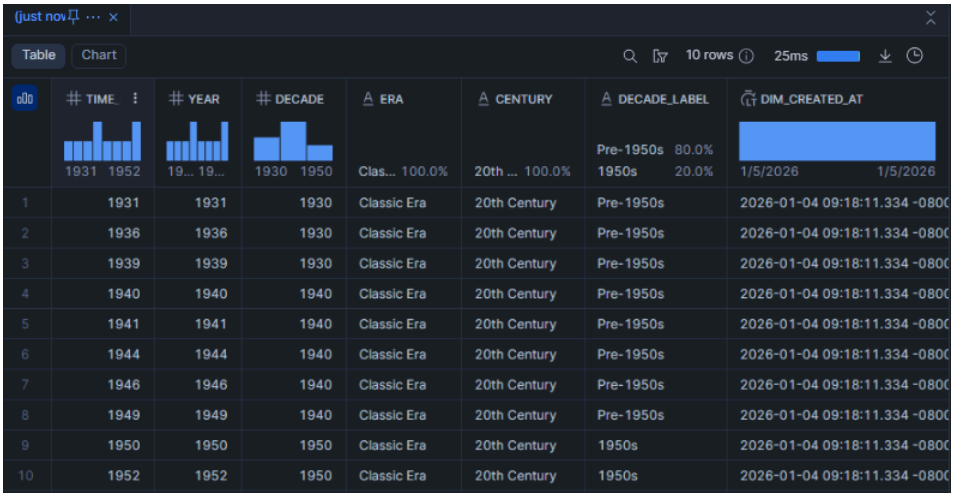
```
-- ============================================================================
-- dim_time
-- ============================================================================
SELECT COUNT(*) FROM dim_time;
```

ผลลัพธ์ที่ได้:




```
SELECT * FROM dim_time ORDER BY year LIMIT 10;
```

ผลลัพธ์ที่ได้:



Dimensions Summary

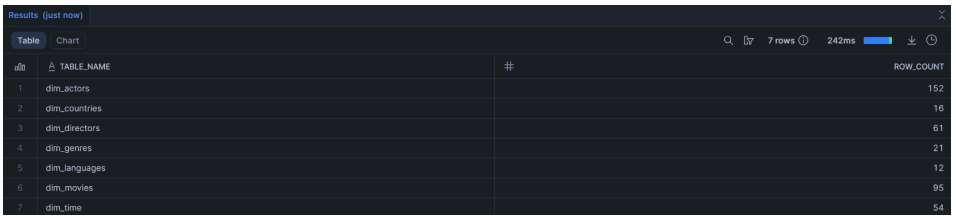
Run All Dimensions:

```
dbt run --select marts.dimensions
```

Verify All:

```
-- ============================================================================
-- Dimensions Summary
-- ============================================================================
-- Check all dimension tables
SELECT 'dim_movies' as table_name, COUNT(*) as row_count FROM dim_movies
UNION ALL
SELECT 'dim_genres', COUNT(*) FROM dim_genres
UNION ALL
SELECT 'dim_directors', COUNT(*) FROM dim_directors
UNION ALL
SELECT 'dim_actors', COUNT(*) FROM dim_actors
UNION ALL
SELECT 'dim_countries', COUNT(*) FROM dim_countries
UNION ALL
SELECT 'dim_languages', COUNT(*) FROM dim_languages
UNION ALL
SELECT 'dim_time', COUNT(*) FROM dim_time
ORDER BY table_name;
```

ผลลัพธ์ที่ได้:



✅Phase 10: Dimensional Modeling (Star Schema) COMPLETE! 🎉

เราได้สร้าง Dimension tables ไป 7 ตาราง ดังนี้:

```
✅dim_movies
✅dim_genres
✅dim_directors
✅dim_actors
✅dim_countries
✅dim_languages
✅dim_time
```