Department of Computer Science and Engineering (IoT and Cyber Security with Block Chain Technology)

T.Y B. Tech, Semester: VI

Academic Year 2024-25

Name: Parth Mehta **SAPID: 60019220096**

Batch: B-2 **Roll No: B045**

Experiment No. 10

Aim: To Create Pipeline for the Gold Layer by using ETL Tools (dbt)

Theory:

=>Medallion Architecture: A Detailed Overview

The medallion architecture is a data design pattern widely used in modern data engineering to organize and process data through progressive layers of refinement. Each layer represents a different stage of data processing, named after precious metals to signify increasing value.

=>Core Layers of the Medallion Architecture

1. Bronze Layer (Raw Layer)

- **Purpose**: Acts as the landing zone for raw data
- **Characteristics**:
    - Contains unmodified source data in its original format
    - Preserves the complete source system information
    - Often includes metadata like ingestion timestamps and source identifiers
    - No business logic or transformations applied
    - Serves as a historical archive of raw data
- **Implementation**: Usually stored as tables with minimal schema enforcement

2. Silver Layer (Cleansed Layer)

- **Purpose**: Standardizes and cleanses data
- **Characteristics**:
    - Contains validated and cleansed data
    - Enforces basic data quality rules
    - Removes duplicates and standardizes formats

- Applies type conversions and basic normalization
- Resolves schema inconsistencies
- May include initial derived fields

- **Implementation**: Often implemented as views or tables with standardized schemas

3. Gold Layer (Business Layer)

- **Purpose**: Provides business-ready data products
- **Characteristics**:
    - Contains aggregated, transformed data optimized for consumption
    - Implements complex business logic and calculations
    - Organizes data into dimensional models, fact tables, or specific domain datasets
    - Optimized for query performance and analytics
    - Ready for direct consumption by BI tools, dashboards, and applications
    - Contains business metrics, KPIs, and domain-specific aggregates
- **Implementation**: Usually materialized as tables optimized for read performance

=>Benefits of the Medallion Architecture

1. **Separation of Concerns**: Each layer has a specific purpose, making the system easier to maintain
2. **Data Lineage**: Clear tracking of how data moves and transforms through the pipeline
3. **Incremental Processing**: Enables efficient processing of only new or changed data
4. **Recovery and Debugging**: Ability to reprocess data from any layer
5. **Isolation of Complexity**: Complex transformations are built progressively
6. **Reusability**: Intermediate layers can be reused for multiple downstream purposes
7. **Quality Control**: Progressive validation and quality checks at each layer

dbt (Data Build Tool)

dbt is an open-source transformation tool that enables data analysts and engineers to transform data in their warehouse using SQL-based transformations.

Key Components of dbt

1. **Models**: SQL files that transform source data into final outputs

2. **Sources**: Configurations that define the external tables dbt will query
3. **Tests**: Assertions about your data to ensure quality and correctness
4. **Documentation**: Automated documentation generation for your data models
5. **Macros**: Reusable SQL snippets (similar to functions)
6. **Seeds**: CSV files that can be loaded as tables
7. **Snapshots**: Point-in-time snapshots of changing data
8. **Materializations**: How models are physically implemented (table, view, etc.)

dbt Architecture and Implementation
Implementation Process

1. **Project Setup**:
   - Initialize a dbt project (dbt init)
   - Configure connection profiles (profiles.yml)
   - Define project settings (dbt_project.yml)
2. **Model Definition**:
   - Create SQL files for transformations
   - Organize models in folder structure
   - Define dependencies between models
3. **Testing and Documentation**:
   - Add data tests
   - Document models and columns
   - Generate documentation
4. **Execution**:
   - Run models (dbt run)
   - Test data (dbt test)
   - Generate documentation (dbt docs generate)

dbt Pipeline Flow

1. **Compilation**: dbt compiles SQL models into executable SQL
2. **Dependency Resolution**: dbt builds a directed acyclic graph (DAG) of dependencies
3. **Execution**: Models are executed in dependency order
4. **Materialization**: Results are materialized according to defined strategies
5. **Testing**: Tests are run against the materialized models
6. **Documentation**: Docs are generated based on model definitions and metadata

dbt Materializations

1. **Table**: Full rebuild of table each run
2. **View**: Creates a SQL view (no data storage)
3. **Incremental**: Processes only new/changed records
4. **Ephemeral**: Not materialized but compiled into dependent models

Implementation of Medallion Architecture with dbt
Project Structure

```
models/
├── bronze/
│   ├── sources.yml      # External data source definitions
│   ├── raw_orders.sql   # Bronze layer raw data models
│   └── raw_customers.sql
├── silver/
│   ├── clean_orders.sql    # Silver layer standardized models
│   └── clean_customers.sql
└── gold/
    ├── customer_metrics.sql  # Gold layer business models
    └── daily_sales.sql
```

Bronze Layer Implementation
The bronze layer in dbt typically connects to source systems and brings in raw data with minimal transformation:

```
sql
-- bronze/raw_orders.sql
{{ config(materialized='table') }}

SELECT
  *,
  CURRENT_TIMESTAMP() as ingestion_timestamp,
  'source_system' as data_source
FROM {{ source('operational_db', 'orders') }}
```

Silver Layer Implementation

The silver layer applies data cleansing and standardization:

```sql
-- silver/clean_orders.sql
{{ config(materialized='view') }}

SELECT
  order_id,
  customer_id,
  TRIM(product) as product,
  COALESCE(amount, 0) as amount,
  STR_TO_DATE(order_date, '%Y-%m-%d') as order_date
FROM {{ ref('raw_orders') }}
WHERE order_id IS NOT NULL
```

Gold Layer Implementation

The gold layer creates business-ready models with metrics and calculations:

```sql
-- gold/customer_metrics.sql
{{ config(materialized='table') }}

WITH customer_orders AS (
  SELECT
    c.customer_id,
    c.customer_name,
    COUNT(o.order_id) AS total_orders,
    SUM(o.amount) AS total_spent,
    MAX(o.order_date) AS last_order_date
  FROM {{ ref('clean_customers') }} c
  LEFT JOIN {{ ref('clean_orders') }} o ON c.customer_id = o.customer_id
  GROUP BY c.customer_id, c.customer_name
)

SELECT
  customer_id,
  customer_name,
  COALESCE(total_orders, 0) AS total_orders,
  COALESCE(total_spent, 0) AS total_spent,
  last_order_date,
  CASE
    WHEN total_spent > 300 THEN 'High Value'
    WHEN total_spent > 100 THEN 'Medium Value'
    ELSE 'Low Value'
  END AS customer_segment
FROM customer_orders
```

Implementation of Data Testing

dbt enables data quality testing at each layer:

```yaml
# models/gold/schema.yml
version: 2

models:
  - name: customer_metrics
    description: "Customer metrics with segmentation"
    columns:
      - name: customer_id
        description: "Unique customer identifier"
        tests:
          - unique
          - not_null
      - name: total_spent
        description: "Total amount spent by customer"
        tests:
          - not_null
```

dbt Pipeline Flow in the Medallion Architecture

1. **Source Connection**: dbt connects to source systems defined in sources.yml
2. **Bronze Layer Creation**: Raw data is loaded into bronze layer models
3. **Silver Layer Transformation**: Bronze data is cleansed and standardized
4. **Gold Layer Aggregation**: Silver data is transformed into business metrics
5. **Testing**: Tests verify data quality at each layer
6. **Documentation**: The entire data pipeline is documented

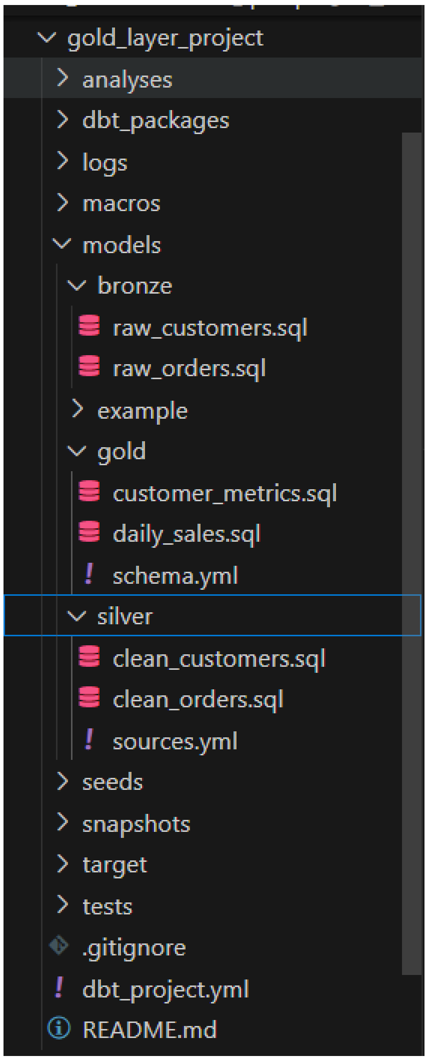
Benefits of Using dbt for Medallion Architecture

1. **Modularity**: Each transformation step is defined in separate SQL files
2. **Version Control**: All transformations are version-controlled
3. **Testing**: Built-in testing ensures data quality
4. **Documentation**: Automated documentation of the entire data pipeline
5. **Dependency Management**: Automatic handling of model dependencies
6. **Incremental Processing**: Efficient processing of new/changed data
7. **Reusability**: Common logic can be extracted into macros
8. **Observability**: Clear visualization of the entire data pipeline

Department of Computer Science and Engineering (IoT and Cyber Security with Block Chain Technology)

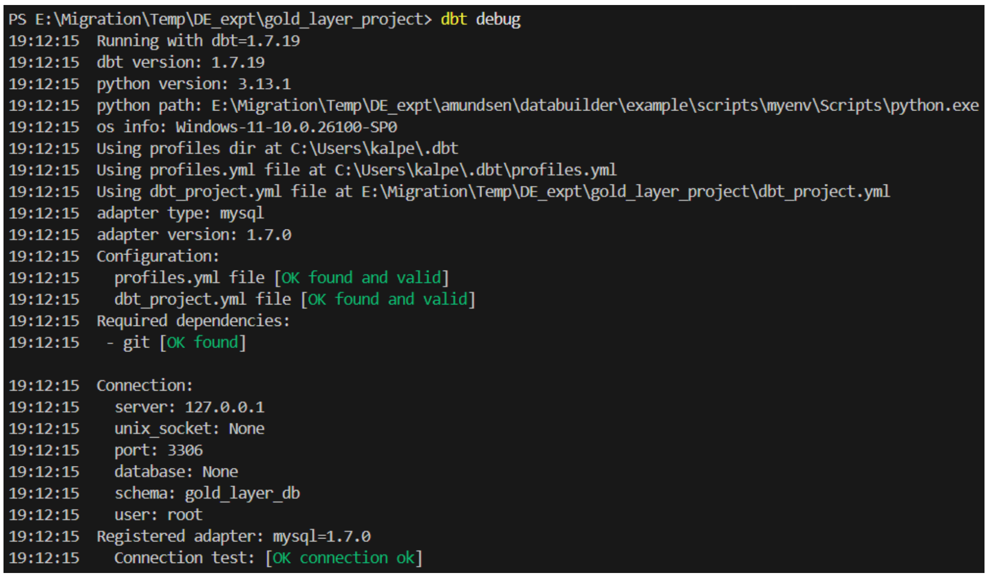
Output:

- **dbt init gold_layer_project**



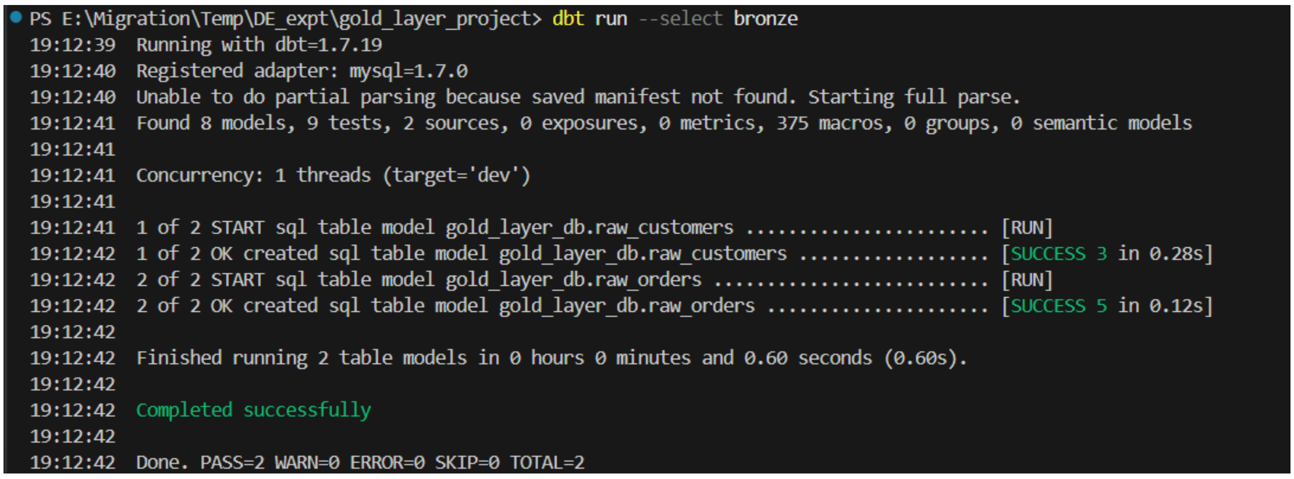
```
gold_layer_project
  analyses
  dbt_packages
  logs
  macros
  models
    bronze
      raw_customers.sql
      raw_orders.sql
    example
    gold
      customer_metrics.sql
      daily_sales.sql
      schema.yml
    silver
      clean_customers.sql
      clean_orders.sql
      sources.yml
  seeds
  snapshots
  target
  tests
  .gitignore
  dbt_project.yml
  README.md
```

- **dbt debug**



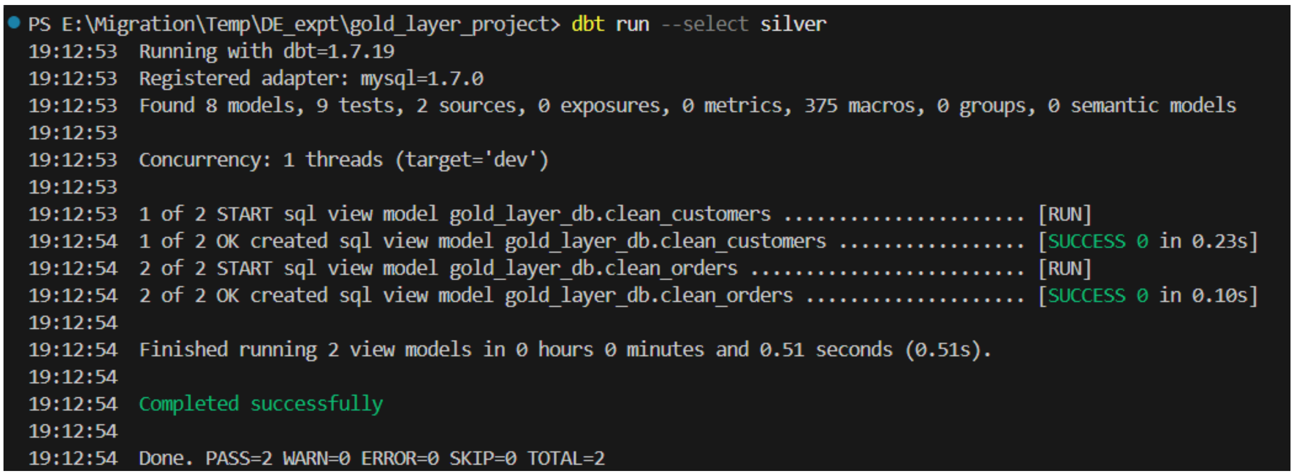
```
PS E:\Migration\Temp\DE_expt\gold_layer_project> dbt debug
19:12:15  Running with dbt=1.7.19
19:12:15  dbt version: 1.7.19
19:12:15  python version: 3.13.1
19:12:15  python path: E:\Migration\Temp\DE_expt\amundsen\databuilder\example\scripts\myenv\Scripts\python.exe
19:12:15  os info: Windows-11-10.0.26100-SP0
19:12:15  Using profiles dir at C:\Users\kalpe\.dbt
19:12:15  Using profiles.yml file at C:\Users\kalpe\.dbt\profiles.yml
19:12:15  Using dbt_project.yml file at E:\Migration\Temp\DE_expt\gold_layer_project\dbt_project.yml
19:12:15  adapter type: mysql
19:12:15  adapter version: 1.7.0
19:12:15  Configuration:
19:12:15    profiles.yml file [OK found and valid]
19:12:15    dbt_project.yml file [OK found and valid]
19:12:15  Required dependencies:
19:12:15   - git [OK found]

19:12:15  Connection:
19:12:15    server: 127.0.0.1
19:12:15    unix_socket: None
19:12:15    port: 3306
19:12:15    database: None
19:12:15    schema: gold_layer_db
19:12:15    user: root
19:12:15  Registered adapter: mysql=1.7.0
19:12:15    Connection test: [OK connection ok]
```

- **dbt run --select bronze**



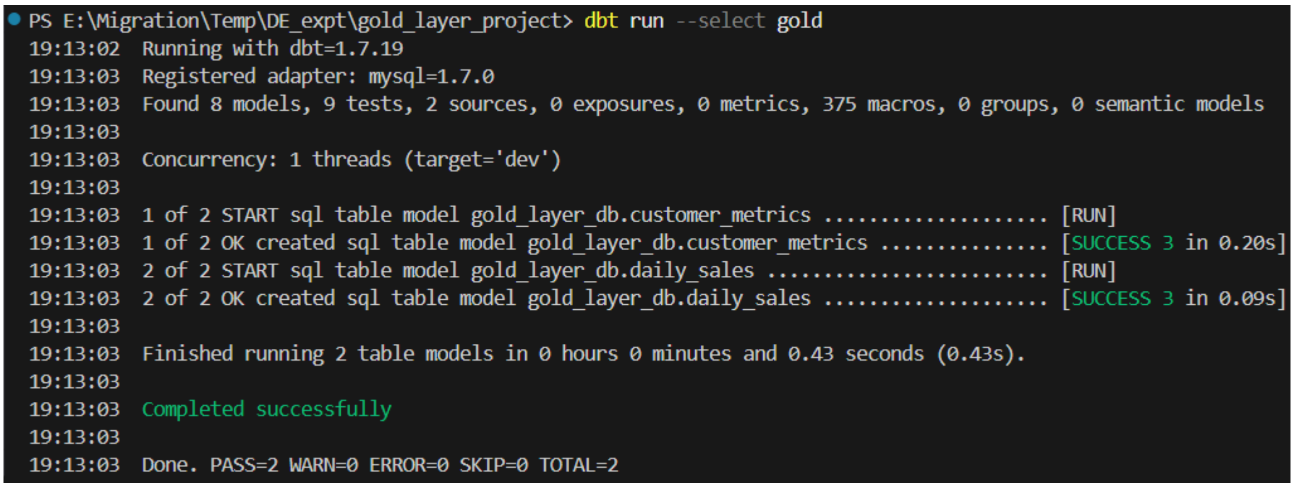
```
PS E:\Migration\Temp\DE_expt\gold_layer_project> dbt run --select bronze
19:12:39  Running with dbt=1.7.19
19:12:40  Registered adapter: mysql=1.7.0
19:12:40  Unable to do partial parsing because saved manifest not found. Starting full parse.
19:12:41  Found 8 models, 9 tests, 2 sources, 0 exposures, 0 metrics, 375 macros, 0 groups, 0 semantic models
19:12:41
19:12:41  Concurrency: 1 threads (target='dev')
19:12:41
19:12:41  1 of 2 START sql table model gold_layer_db.raw_customers ...................... [RUN]
19:12:42  1 of 2 OK created sql table model gold_layer_db.raw_customers ................. [SUCCESS 3 in 0.28s]
19:12:42  2 of 2 START sql table model gold_layer_db.raw_orders ......................... [RUN]
19:12:42  2 of 2 OK created sql table model gold_layer_db.raw_orders .................... [SUCCESS 5 in 0.12s]
19:12:42
19:12:42  Finished running 2 table models in 0 hours 0 minutes and 0.60 seconds (0.60s).
19:12:42
19:12:42  Completed successfully
19:12:42
19:12:42  Done. PASS=2 WARN=0 ERROR=0 SKIP=0 TOTAL=2
```

- **dbt run --select silver**



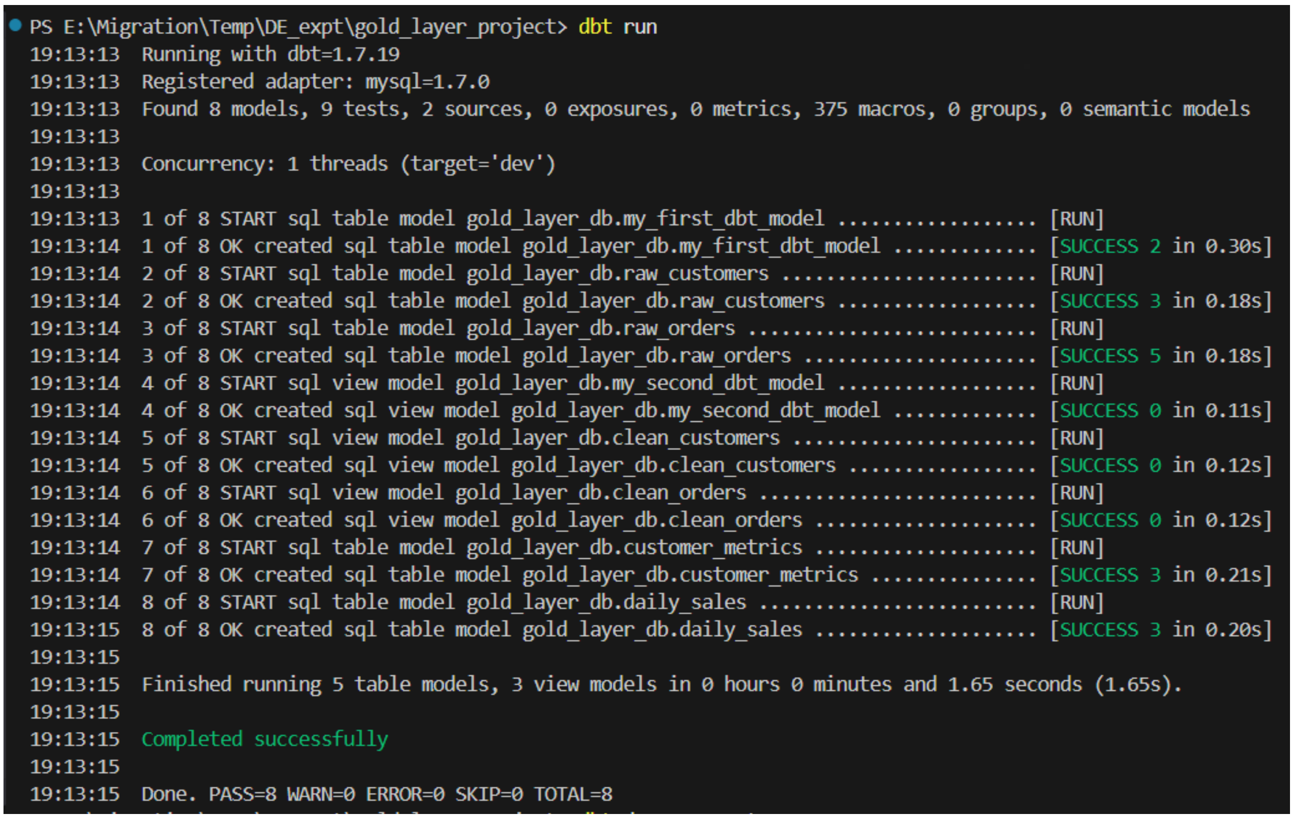
```
PS E:\Migration\Temp\DE_expt\gold_layer_project> dbt run --select silver
19:12:53  Running with dbt=1.7.19
19:12:53  Registered adapter: mysql=1.7.0
19:12:53  Found 8 models, 9 tests, 2 sources, 0 exposures, 0 metrics, 375 macros, 0 groups, 0 semantic models
19:12:53
19:12:53  Concurrency: 1 threads (target='dev')
19:12:53
19:12:53  1 of 2 START sql view model gold_layer_db.clean_customers ..................... [RUN]
19:12:54  1 of 2 OK created sql view model gold_layer_db.clean_customers ................ [SUCCESS 0 in 0.23s]
19:12:54  2 of 2 START sql view model gold_layer_db.clean_orders ........................ [RUN]
19:12:54  2 of 2 OK created sql view model gold_layer_db.clean_orders ................... [SUCCESS 0 in 0.10s]
19:12:54
19:12:54  Finished running 2 view models in 0 hours 0 minutes and 0.51 seconds (0.51s).
19:12:54
19:12:54  Completed successfully
19:12:54
19:12:54  Done. PASS=2 WARN=0 ERROR=0 SKIP=0 TOTAL=2
```

- **dbt run --select gold**



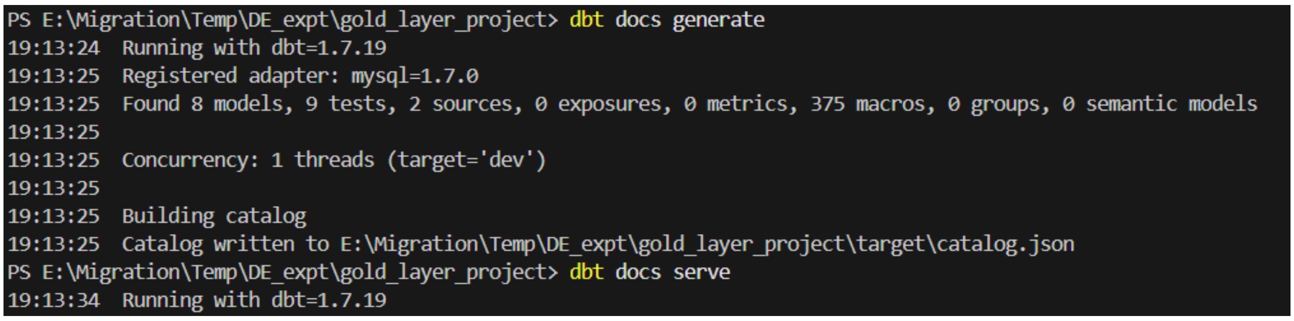
```
PS E:\Migration\Temp\DE_expt\gold_layer_project> dbt run --select gold
19:13:02  Running with dbt=1.7.19
19:13:03  Registered adapter: mysql=1.7.0
19:13:03  Found 8 models, 9 tests, 2 sources, 0 exposures, 0 metrics, 375 macros, 0 groups, 0 semantic models
19:13:03
19:13:03  Concurrency: 1 threads (target='dev')
19:13:03
19:13:03  1 of 2 START sql table model gold_layer_db.customer_metrics ................... [RUN]
19:13:03  1 of 2 OK created sql table model gold_layer_db.customer_metrics .............. [SUCCESS 3 in 0.20s]
19:13:03  2 of 2 START sql table model gold_layer_db.daily_sales ........................ [RUN]
19:13:03  2 of 2 OK created sql table model gold_layer_db.daily_sales ................... [SUCCESS 3 in 0.09s]
19:13:03
19:13:03  Finished running 2 table models in 0 hours 0 minutes and 0.43 seconds (0.43s).
19:13:03
19:13:03  Completed successfully
19:13:03
19:13:03  Done. PASS=2 WARN=0 ERROR=0 SKIP=0 TOTAL=2
```

- **dbt run**



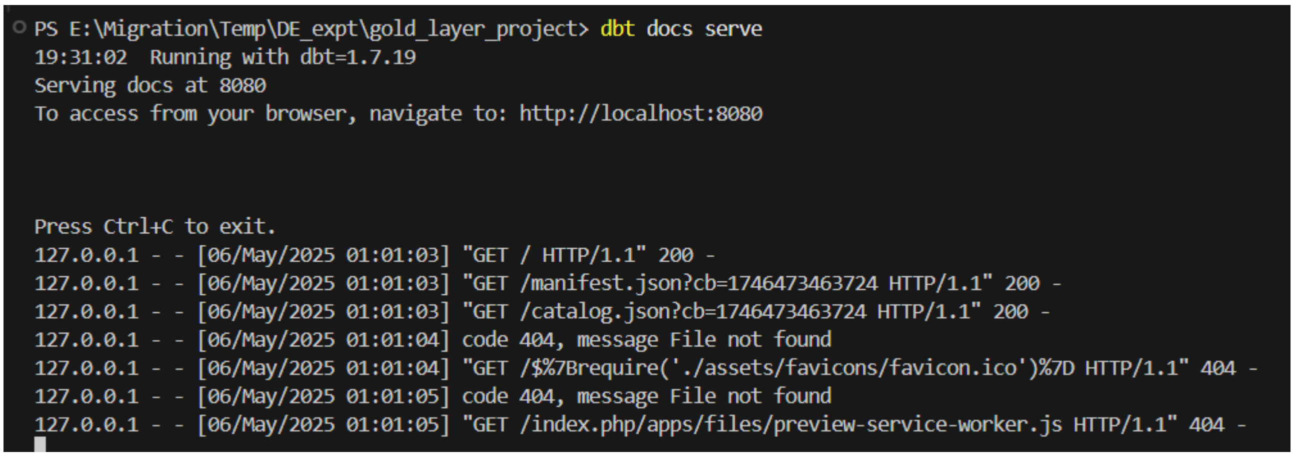
```
PS E:\Migration\Temp\DE_expt\gold_layer_project> dbt run
19:13:13  Running with dbt=1.7.19
19:13:13  Registered adapter: mysql=1.7.0
19:13:13  Found 8 models, 9 tests, 2 sources, 0 exposures, 0 metrics, 375 macros, 0 groups, 0 semantic models
19:13:13
19:13:13  Concurrency: 1 threads (target='dev')
19:13:13
19:13:13  1 of 8 START sql table model gold_layer_db.my_first_dbt_model .................. [RUN]
19:13:14  1 of 8 OK created sql table model gold_layer_db.my_first_dbt_model ............. [SUCCESS 2 in 0.30s]
19:13:14  2 of 8 START sql table model gold_layer_db.raw_customers ....................... [RUN]
19:13:14  2 of 8 OK created sql table model gold_layer_db.raw_customers .................. [SUCCESS 3 in 0.18s]
19:13:14  3 of 8 START sql table model gold_layer_db.raw_orders .......................... [RUN]
19:13:14  3 of 8 OK created sql table model gold_layer_db.raw_orders ..................... [SUCCESS 5 in 0.18s]
19:13:14  4 of 8 START sql view model gold_layer_db.my_second_dbt_model .................. [RUN]
19:13:14  4 of 8 OK created sql view model gold_layer_db.my_second_dbt_model ............. [SUCCESS 0 in 0.11s]
19:13:14  5 of 8 START sql view model gold_layer_db.clean_customers ...................... [RUN]
19:13:14  5 of 8 OK created sql view model gold_layer_db.clean_customers ................. [SUCCESS 0 in 0.12s]
19:13:14  6 of 8 START sql view model gold_layer_db.clean_orders ......................... [RUN]
19:13:14  6 of 8 OK created sql view model gold_layer_db.clean_orders .................... [SUCCESS 0 in 0.12s]
19:13:14  7 of 8 START sql table model gold_layer_db.customer_metrics .................... [RUN]
19:13:14  7 of 8 OK created sql table model gold_layer_db.customer_metrics ............... [SUCCESS 3 in 0.21s]
19:13:14  8 of 8 START sql table model gold_layer_db.daily_sales ......................... [RUN]
19:13:15  8 of 8 OK created sql table model gold_layer_db.daily_sales .................... [SUCCESS 3 in 0.20s]
19:13:15
19:13:15  Finished running 5 table models, 3 view models in 0 hours 0 minutes and 1.65 seconds (1.65s).
19:13:15
19:13:15  Completed successfully
19:13:15
19:13:15  Done. PASS=8 WARN=0 ERROR=0 SKIP=0 TOTAL=8
```

- **dbt docs generate**



```
PS E:\Migration\Temp\DE_expt\gold_layer_project> dbt docs generate
19:13:24  Running with dbt=1.7.19
19:13:25  Registered adapter: mysql=1.7.0
19:13:25  Found 8 models, 9 tests, 2 sources, 0 exposures, 0 metrics, 375 macros, 0 groups, 0 semantic models
19:13:25
19:13:25  Concurrency: 1 threads (target='dev')
19:13:25
19:13:25  Building catalog
19:13:25  Catalog written to E:\Migration\Temp\DE_expt\gold_layer_project\target\catalog.json
PS E:\Migration\Temp\DE_expt\gold_layer_project> dbt docs serve
19:13:34  Running with dbt=1.7.19
```

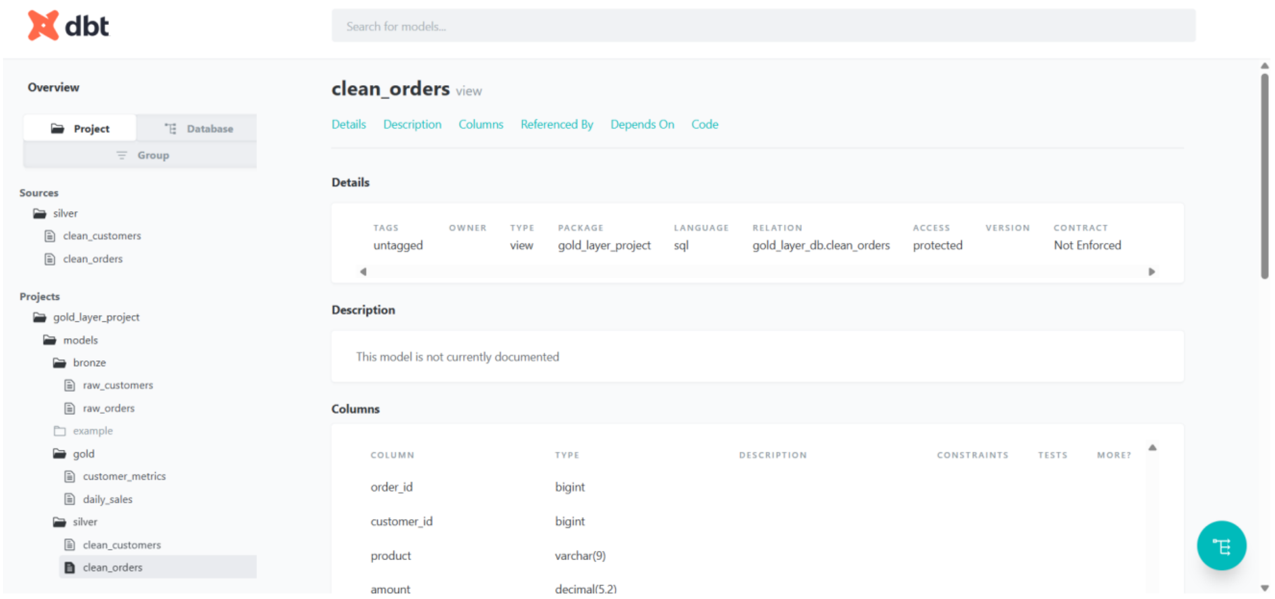
- **dbt docs serve**



```
PS E:\Migration\Temp\DE_expt\gold_layer_project> dbt docs serve
19:31:02  Running with dbt=1.7.19
Serving docs at 8080
To access from your browser, navigate to: http://localhost:8080



Press Ctrl+C to exit.
127.0.0.1 - - [06/May/2025 01:01:03] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [06/May/2025 01:01:03] "GET /manifest.json?cb=1746473463724 HTTP/1.1" 200 -
127.0.0.1 - - [06/May/2025 01:01:03] "GET /catalog.json?cb=1746473463724 HTTP/1.1" 200 -
127.0.0.1 - - [06/May/2025 01:01:04] code 404, message File not found
127.0.0.1 - - [06/May/2025 01:01:04] "GET /$%7Brequire('./assets/favicons/favicon.ico')%7D HTTP/1.1" 404 -
127.0.0.1 - - [06/May/2025 01:01:05] code 404, message File not found
127.0.0.1 - - [06/May/2025 01:01:05] "GET /index.php/apps/files/preview-service-worker.js HTTP/1.1" 404 -
```

OUTPUT:

dbt tool:



clean_orders view

Details | Description | Columns | Referenced By | Depends On | Code

Details

TAGS	OWNER	TYPE	PACKAGE	LANGUAGE	RELATION	ACCESS	VERSION	CONTRACT
untagged		view	gold_layer_project	sql	gold_layer_db.clean_orders	protected		Not Enforced

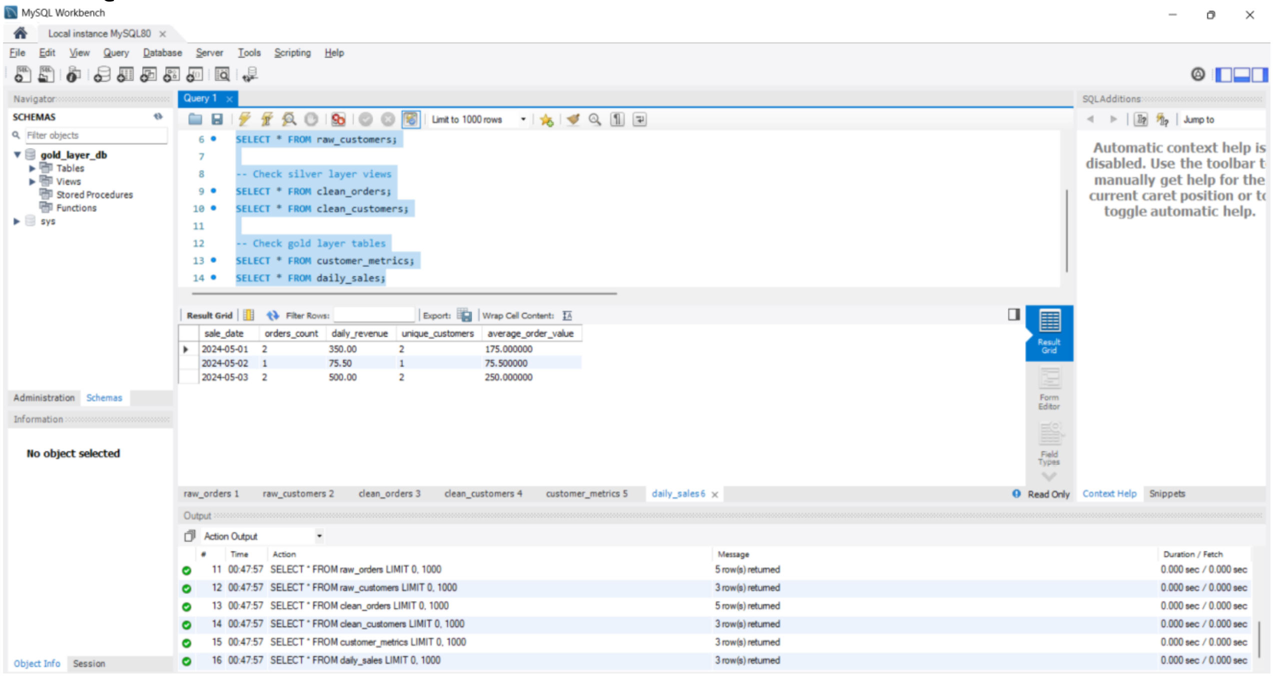
Description

This model is not currently documented

Columns

COLUMN	TYPE	DESCRIPTION	CONSTRAINTS	TESTS	MORE?
order_id	bigint				
customer_id	bigint				
product	varchar(9)				
amount	decimal(5,2)				

MYSQL Workbench:



```
SELECT * FROM raw_customers;

-- Check silver layer views
SELECT * FROM clean_orders;
SELECT * FROM clean_customers;

-- Check gold layer tables
SELECT * FROM customer_metrics;
SELECT * FROM daily_sales;
```

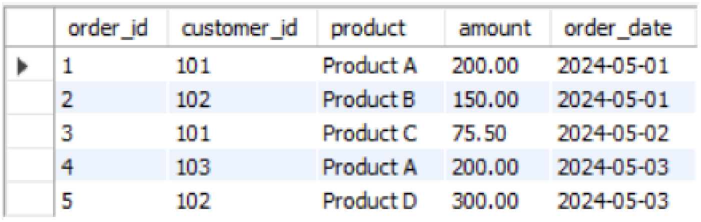
	sale_date	orders_count	daily_revenue	unique_customers	average_order_value
▶	2024-05-01	2	350.00	2	175.000000
	2024-05-02	1	75.50	1	75.500000
	2024-05-03	2	500.00	2	250.000000

```
#   Time      Action                                          Message            Duration / Fetch
11  00:47:57  SELECT * FROM raw_orders LIMIT 0, 1000          5 row(s) returned  0.000 sec / 0.000 sec
12  00:47:57  SELECT * FROM raw_customers LIMIT 0, 1000       3 row(s) returned  0.000 sec / 0.000 sec
13  00:47:57  SELECT * FROM clean_orders LIMIT 0, 1000        5 row(s) returned  0.000 sec / 0.000 sec
14  00:47:57  SELECT * FROM clean_customers LIMIT 0, 1000     3 row(s) returned  0.000 sec / 0.000 sec
15  00:47:57  SELECT * FROM customer_metrics LIMIT 0, 1000    3 row(s) returned  0.000 sec / 0.000 sec
16  00:47:57  SELECT * FROM daily_sales LIMIT 0, 1000         3 row(s) returned  0.000 sec / 0.000 sec
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

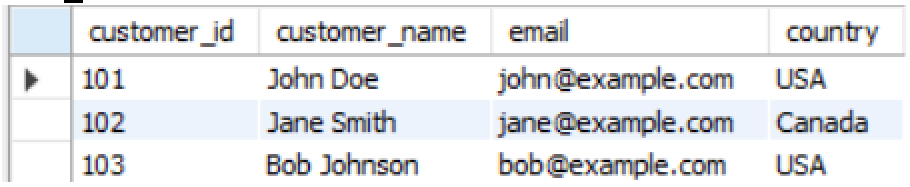
Bronze layer:

raw_orders



order_id	customer_id	product	amount	order_date
1	101	Product A	200.00	2024-05-01
2	102	Product B	150.00	2024-05-01
3	101	Product C	75.50	2024-05-02
4	103	Product A	200.00	2024-05-03
5	102	Product D	300.00	2024-05-03

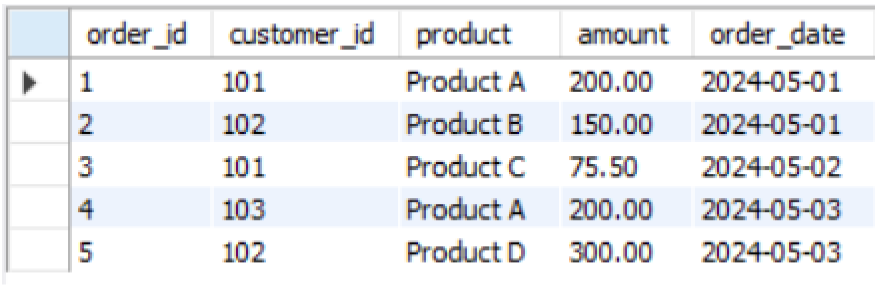
raw_customers



customer_id	customer_name	email	country
101	John Doe	john@example.com	USA
102	Jane Smith	jane@example.com	Canada
103	Bob Johnson	bob@example.com	USA

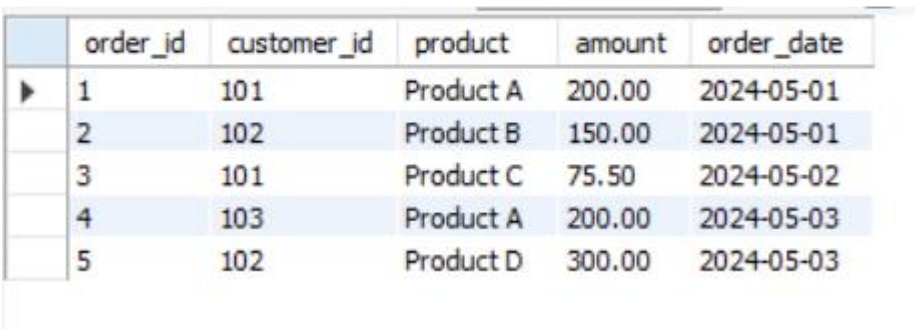
Silver Layer:

clean_orders



order_id	customer_id	product	amount	order_date
1	101	Product A	200.00	2024-05-01
2	102	Product B	150.00	2024-05-01
3	101	Product C	75.50	2024-05-02
4	103	Product A	200.00	2024-05-03
5	102	Product D	300.00	2024-05-03

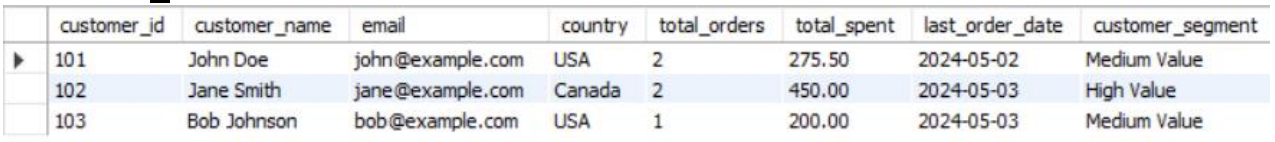
clean_customers



order_id	customer_id	product	amount	order_date
1	101	Product A	200.00	2024-05-01
2	102	Product B	150.00	2024-05-01
3	101	Product C	75.50	2024-05-02
4	103	Product A	200.00	2024-05-03
5	102	Product D	300.00	2024-05-03

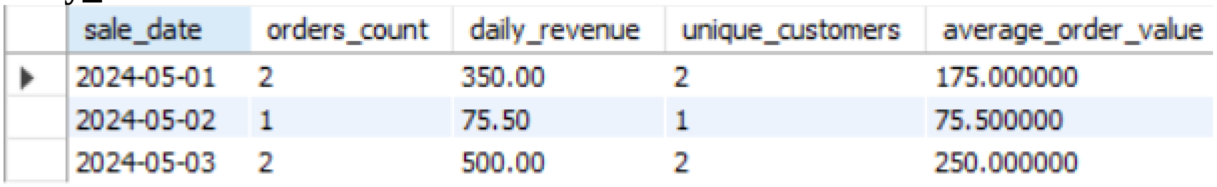
Gold Layer:

customer_metrics



customer_id	customer_name	email	country	total_orders	total_spent	last_order_date	customer_segment
101	John Doe	john@example.com	USA	2	275.50	2024-05-02	Medium Value
102	Jane Smith	jane@example.com	Canada	2	450.00	2024-05-03	High Value
103	Bob Johnson	bob@example.com	USA	1	200.00	2024-05-03	Medium Value

daily_sales



sale_date	orders_count	daily_revenue	unique_customers	average_order_value
2024-05-01	2	350.00	2	175.000000
2024-05-02	1	75.50	1	75.500000
2024-05-03	2	500.00	2	250.000000

Conclusion: In conclusion, the medallion architecture implemented with dbt creates a well-structured, maintainable data pipeline that progressively refines data from raw formats to business-ready insights. This approach addresses common data engineering challenges by separating concerns, maintaining data lineage, ensuring quality, and optimizing for both development efficiency and analytical performance.