

Module 3 Lab 1: Data Lifecycle Design and Management

Estimated Time: 60–90 minutes

Risk Level: Moderate

Environment Stress: Low (dbt + Postgres only)

Introduction

In this lab, you will operate and analyze a complete data lifecycle using dbt. Rather than creating new models, you will run, observe, and reason about how data moves through lifecycle stages; from raw ingestion to analytics-ready outputs and how governance, quality, and access decisions apply at each stage. This lab emphasizes understanding and interpretation, not configuration or setup.

What You Have Already Done (Labs 1–3)

Before starting M3 Lab 1, you have already:

- Installed and validated PostgreSQL and dbt (M2 Lab 1)
- Explored raw data and documented governance classifications (M2 Lab 2)
- Built and validated dbt models across staging → core → marts (M2 Lab 3)

👉 **M3 Lab 1 does NOT repeat these tasks.**

Instead, it focuses on lifecycle execution, lineage visualization, and decision documentation.

Learning Outcomes

By completing this lab, you will be able to:

1. Map the full data lifecycle: raw → staging → core → marts
2. Execute a controlled lifecycle build using dbt
3. Generate and interpret a dbt lineage (DAG) graph
4. Explain how lifecycle stages support:
 - Data quality
 - Governance
 - Access control
 - Analytics readiness
5. Document lifecycle decisions using a structured framework

Lab Concept: Data Lifecycle in Practice

A well-designed data lifecycle ensures that:

- Raw data is preserved and auditable
- Staging models standardize and clean data
- Core models enforce business meaning and integrity
- Marts serve specific operational or analytical use cases

dbt makes this lifecycle visible and testable, especially through its lineage graph.

Before You Begin

Make sure that:

- You successfully completed M2 Labs 1–3
 - You are working inside your cloned course repository
 - You are using the same database credentials configured in Lab 1
 - Your PostgreSQL service is running
- ⚠ If M2 Lab 3 did not run successfully, fix that first before continuing.

Step 1: Run the Full Lifecycle Pipeline

1. Start your VM
2. Navigate your clone course repo root

```
cd ~/IT4065C-Labs
```

```
kofi@18ntiikxil02:/mnt/c/Users/ntiik$ cd ~/IT4065C-Labs
kofi@18ntiikxil02:~/IT4065C-Labs$
```

3. From the root of the course repository, run
bash labs/module_3/lab4/_run_all.sh

Use the same database credentials configured in M2 Lab 1 (do not change them).

This script will:

1. Validate your dbt environment
2. Confirm raw data availability
3. Run all Lab 3 dbt models (**staging → core → marts**)
4. Execute dbt tests
5. Generate dbt documentation for lineage analysis

 This may take a few minutes.

If the script completes without errors, you are ready to continue. A successful run shows display “Completed Successfully”, see images below.

```
ps: kofi@18ntiikxil02:~/IT4065C- x + v
kofi@18ntiikxil02:~/IT4065C-Labs$ bash labs/module_3/lab4/_run_all.sh
--- Lab 4: Lifecycle Build + Test + Docs (Module 3) ---

Step 1/4: dbt debug
15:32:32 Running with dbt=1.1.2
15:32:32 dbt version: 1.1.2
15:32:32 python version: 3.10.12
15:32:32 python path: /home/kofi/IT4065C-Labs/dbt-venv/bin/python3
15:32:32 os info: Linux-5.15.167.4-microsoft-standard-WSL2-x86_64-with-glibc2.35
15:32:33 Using profiles dir at /home/kofi/IT4065C-Labs/dbt/it4065c_platform
15:32:33 Using profiles.yml file at /home/kofi/IT4065C-Labs/dbt/it4065c_platform/dbt_profiles.yml
15:32:33 Using dbt_project.yml file at /home/kofi/IT4065C-Labs/dbt/it4065c_platform/dbt_project.yml
15:32:33 adapter type: postgres
15:32:33 adapter version: 1.10.0
15:32:33 Configuration:
15:32:33   profiles.yml file [OK found and valid]
15:32:33   dbt_project.yml file [OK found and valid]
15:32:33 Required dependencies:
15:32:33   - git [OK found]
15:32:33 Connection:
15:32:33   host: localhost
15:32:33   port: 5432
15:32:33   user: kofi
15:32:33   database: it4065c
15:32:33   schema: student_kofi
15:32:33   connect_timeout: 10
15:32:33   role: None
15:32:33   search_path: None
15:32:33   keepalives_idle: 0
15:32:33   sslmode: prefer
15:32:33   sslcert: None
15:32:33   sslkey: None
15:32:33   sslrootcert: None
15:32:33   application_name: dbt
15:32:33   retries: 1
15:32:33 Registered adapter: postgres=1.10.0
15:32:33 Connection test: [OK connection ok]

15:32:33 All checks passed!

Step 2/4: Ensure raw tables exist (seed if missing)
Password for user postgres:
--- Checking raw schema tables ---
  List of relations
 Schema |    Name     | Type | Owner
-----+-----+-----+
 raw   | customers | table | postgres
 raw   | order_items | table | postgres
 raw   | orders | table | postgres
 raw   | products | table | postgres
(4 rows)

--- If you do NOT see raw tables above, run this from repo root:
psql -h localhost -U postgres & it4065c -f labs/module_3/lab2_seed.sql
  Then re-run lab4/_run_all.sh ---

Step 3/4: dbt build (Lab 3 models = staging->core->marts + tests)
15:34:53 Registered adapter: postgres=1.10.0
15:34:53 Found 10 models, 28 data tests, 464 macros
15:34:54 Concurrency: 2 threads (target='dev')
15:34:54 1 of 28 START test not_null_stg_customers_customer_id ..... [RUN]
15:34:54 2 of 28 START test not_null_stg_order_items_order_id ..... [RUN]
15:34:54 3 of 28 PASS not_null_stg_order_items_order_id ..... [PASS in 0.12s]
15:34:54 4 of 28 START test not_null_stg_order_items_order_id ..... [RUN]
15:34:54 5 of 28 PASS not_null_stg_order_items_order_id ..... [PASS in 0.13s]
15:34:54 6 of 28 START test not_null_stg_order_items_product_id ..... [RUN]
15:34:54 7 of 28 PASS not_null_stg_order_items_product_id ..... [PASS in 0.06s]
15:34:54 8 of 28 START test not_null_stg_order_items_product_id ..... [RUN]
15:34:54 9 of 28 PASS not_null_stg_order_items_product_id ..... [PASS in 0.06s]
15:34:54 10 of 28 START test relationships_stg_order_items_order_id_order_id_ref_stg_orders_ ..... [RUN]
15:34:54 11 of 28 PASS relationships_stg_order_items_order_id_order_id_ref_stg_orders_ ..... [PASS in 0.07s]
15:34:54 12 of 28 START test unique_stg_order_items_order_id ..... [RUN]
15:34:55 13 of 28 PASS unique_stg_order_items_order_id ..... [PASS in 0.08s]
15:34:55 14 of 28 PASS unique_stg_order_items_order_id ..... [PASS in 0.07s]
15:34:55 15 of 28 START test unique_stg_products_product_id ..... [RUN]
```

```

15:34:55 16 of 28 START test_unique_dim_customers_customers_id ..... [RUN]
15:34:55 16 of 28 PASS not_null_dim_customers_customers_id ..... [PASS] in 0.06s
15:34:55 17 of 28 START test_not_null_fct_order_items_order_id ..... [RUN]
15:34:55 17 of 28 PASS not_null_fct_order_items_order_id ..... [PASS] in 0.07s
15:34:55 18 of 28 START test_not_null_fct_order_items_order_id ..... [RUN]
15:34:55 18 of 28 PASS not_null_fct_order_items_order_id ..... [PASS] in 0.07s
15:34:55 19 of 28 START test_not_null_fct_order_items_product_id ..... [RUN]
15:34:55 19 of 28 PASS not_null_fct_order_items_product_id ..... [PASS] in 0.07s
15:34:55 20 of 28 START test_not_null_fct_order_items_order_id ..... [RUN]
15:34:55 20 of 28 PASS not_null_fct_order_items_order_id ..... [PASS] in 0.07s
15:34:55 21 of 28 START test_not_null_fct_orders_customer_id ..... [RUN]
15:34:55 21 of 28 PASS unique_fct_order_items_order_id ..... [PASS] in 0.08s
15:34:55 22 of 28 START test_not_null_fct_orders_order_id ..... [RUN]
15:34:55 22 of 28 PASS not_null_fct_orders_order_id ..... [PASS] in 0.07s
15:34:55 23 of 28 START test_relationships_fct_order_items_order_id_ref_fct_orders ..... [RUN]
15:34:55 23 of 28 PASS not_null_fct_orders_order_id ..... [PASS] in 0.08s
15:34:55 24 of 28 START test_relationships_fct_order_items_order_id_ref_dim_customers ..... [RUN]
15:34:55 24 of 28 PASS not_null_fct_orders_order_id ..... [PASS] in 0.08s
15:34:55 25 of 28 START test_relationships_fct_order_items_order_id_ref_fct_orders ..... [RUN]
15:34:55 25 of 28 PASS not_null_fct_order_items_order_id_ref_fct_orders ..... [PASS] in 0.10s
15:34:55 26 of 28 START test unique_fct_orders_order_id ..... [RUN]
15:34:55 26 of 28 PASS unique_fct_orders_order_id ..... [PASS] in 0.10s
15:34:55 27 of 28 START test_relationships_fct_order_items_product_id_product_id_ref_dim_products ..... [RUN]
15:34:55 27 of 28 PASS not_null_dim_products_product_id ..... [PASS] in 0.10s
15:34:55 28 of 28 START test_unique_dim_products_product_id ..... [RUN]
15:34:55 28 of 28 PASS unique_dim_products_product_id ..... [PASS] in 0.09s
15:34:55 29 of 28 PASS unique_dim_products_product_id ..... [PASS] in 0.09s
15:34:55 Finished running 28 data tests in 0 hours 0 minutes and 1.57 seconds (1.57s).
15:34:55
15:34:55 Completed successfully
15:34:55
15:34:55 Done. PASS=28 WARN=0 ERROR=0 SKIP=0 NO_OP=0 TOTAL=28
Step 4/4: dbt docs generate (for lineage + documentation)
15:34:59 Running with dbt=1.11.2
15:35:00 Registered adapter: postgres+1.10.0
15:35:00 Found 10 models, 28 data tests, 464 macros
15:35:00
15:35:00 Concurrency: 2 threads (target='dev')
15:35:00
15:35:01 Building catalog
15:35:01 Catalog written to /home/kofi/IT4065C-Labs/dbt/it4065c_platform/target/catalog.json
DONE.
Next: run 'dbt docs serve' and take lineage screenshots per README.
kofi@18ntiikxil02:~/IT4065C-Labs$
```

Step 2: Launch dbt Docs and View Lineage

In this step, you will launch dbt Docs and confirm that the data lineage graph is available.

Step 2.1: Start dbt Docs

1. Navigate to the repository root directory.
2. Run the following commands in order:
`source ~/IT4065C-Labs/dbt-venv/bin/activate`
`cd ~/IT4065C-Labs/dbt/it4065c_platform`
`dbt docs serve --port 8080`
3. You should see output indicating that dbt Docs is running.

```

kofi@18ntiikxil02:~/IT4065C-Labs$ source ~/IT4065C-Labs/dbt-venv/bin/activate
(dbt-venv) kofi@18ntiikxil02:~/IT4065C-Labs$ cd ~/IT4065C-Labs/dbt/it4065c_platform
(dbt-venv) kofi@18ntiikxil02:~/IT4065C-Labs/dbt/it4065c_platform$ dbt docs serve --port 8080
15:51:02 Running with dbt=1.11.2
Serving docs at 8080
To access from your browser, navigate to: http://localhost:8080
[redacted]

Press Ctrl+C to exit.
|
```

4. Open the URL displayed in your browser (usually <http://localhost:8080>).
5. Leave this terminal window running while you work in the browser.

At this point, the dbt Docs interface should open successfully.

Step 3: Explore and Interpret the Data Lineage Graph

In this step, you will use the lineage graph (DAG) to understand how data flows through the lifecycle.

Step 3.1: Open the Lineage Graph

Inside the dbt Docs interface:

1. Click **Project** in the left sidebar.
2. Locate any model under `models/marts/lab3` (for example, `olap_sales_by_day`).
3. From this model page, click the green graph icon in the bottom-right corner to open the lineage graph.

You should now see a lineage graph showing data flowing from `raw` → `staging` → `core` → `marts`.



This initial lineage view shows a model's direct dependencies only. You will explore upstream lifecycle layers in the next steps.

Step 3.2: Verify Mart Dependencies Using the Lineage Graph

1. In dbt Docs, open the model `oltp_order_detail`.
2. Click the lineage graph icon (bottom-right).
3. Observe how the model depends on:
 - o Core fact and dimension models
 - o Upstream staging models
 - o Raw source tables

Screenshot 1: Lifecycle via `oltp_order_detail`

Must show:

- `oltp_order_detail`
- At least one upstream fact or dimension model
- At least one upstream staging model
- At least one upstream raw source

This screenshot demonstrates a complete lifecycle path.



Step 3.3: Confirm Raw → Staging Relationship in Code

1. In dbt Docs, open the `it4065c_platform/models/staging/lab3/stg_orders`.
2. Click Code → Source.
3. Locate the source reference:

```
from {{ source('raw', 'orders') }}
```

Screenshot 2: Source verification (`{{ source() }}`)

- Must show:
 - o `stg_orders`
 - o Code → Source view
 - o The `source('raw', 'orders')` reference

```

8 - Uses {{ source('raw', 'orders') }} so dbt draws RAW + STG edges in the DAG.
9 - Standardizes types and trims strings to reduce downstream errors.
10 -----
11 */
12 with src as
13
14   select
15     *
16   from {{ source('raw', 'orders') }}
17
18 ),
19   standardized as (
20
21   select
22
23     cast(order_id as integer) as order_id,
24     cast(order_qty as integer) as order_qty

```

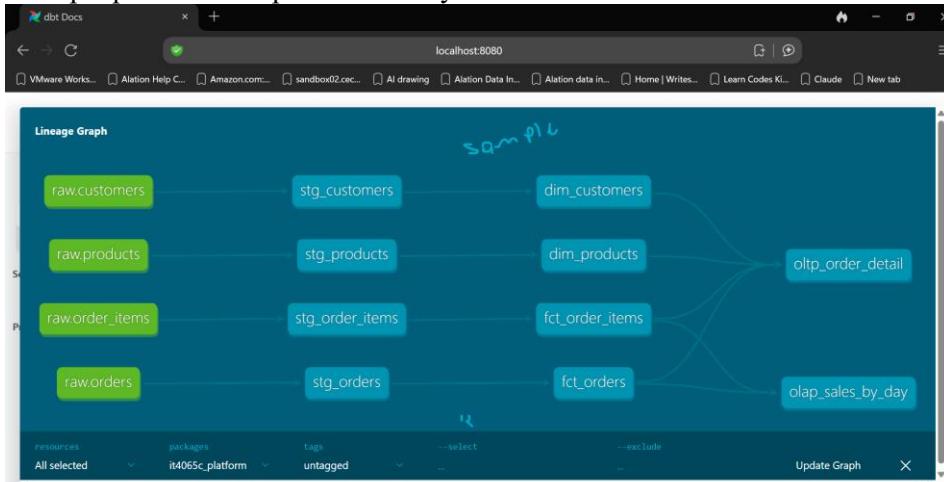
Step 3.4: Write the Data Lifecycle Chain (Short Answer)

In this step, you will use the full lineage graph in dbt Docs to demonstrate your understanding of how data moves through the system.

1. In dbt Docs, click it4065c_platform under project in the left sidebar.
2. Click the lineage graph icon (bottom-right corner) to open the full lineage graph.
3. Observe how data flows from raw sources, through staging and core models, and into mart models.

Screenshot 3: Full project lineage graph

You will see multiple possible data paths. You only need to choose one.



Written Submission

Using the lineage graph, write one complete lifecycle chain that shows how data flows from a raw source to a mart model.

Your lifecycle chain must:

- Start with a raw.* table
- Include one stg_* model
- Include one core model (fct_* or dim_*)
- End with a mart model (olap_* or oltp_*)

Use exact model names as shown in dbt Docs, and use arrows (→) to show data flow.

Example (One Valid Answer)

raw.orders

→ stg_orders

→ fct_orders

→ olap_sales_by_day

Note: You do not need to describe the entire graph.

One correct lifecycle path is sufficient.

Why This Matters

Being able to identify and describe a complete lifecycle chain demonstrates that you understand:

- Where data originates
- How it is transformed
- Where business logic is applied
- Which models are safe for analytics and reporting

This skill is essential for data governance and analytics engineering.