M2 Lab 2: Raw Data Exploration & Data Classification

Estimated Time: 45–75 minutes
Risk Level: Low
Environment Stress: Minimal (Postgres + SQL only)

What You Will Learn After This Lab

By the end of this lab, you will be able to:

- Navigate a real database schema and inspect tables and columns
- Identify PII, financial, and operational data using SQL
- Classify data fields using governance categories (Public, Internal, Sensitive, Restricted)
- Document governance decisions in a structured, auditable format
- Explain *why* certain data requires stricter controls than others

This lab focuses on seeing and thinking, not transforming data.

Important Scope Reminder (Please Read)

- ❌ No dbt in this lab
- ❌ No data transformations
- ❌ No performance tuning
- ✅ Yes to exploration, classification, and documentation

If you feel unsure at any point, that is normal.
You are learning how professionals reason about data, not memorizing steps.

What You Are Given

When you cloned the course repository in Lab 1, you already received:

- A working Postgres database (it4065c)
- A raw schema containing retail data
- SQL files located at:
  - *IT4065C-Labs/labs/module_2/lab2_seed.sql*
  - *IT4065C-Labs/labs/module_2/ lab2_insert_templates.sql*
  - *IT4065C-Labs/labs/module_2/ lab2_governance_register.sql*
  - *IT4065C-Labs/labs/module_2/ lab2_verify_register.sql*

You will use these files only for this Lab.

Step 1: Resume Your Environment (Low Stress Check)

- If you restarted your VM:
  cd ~/IT4065C-Labs
- Then connect to Postgres:
  psql -h localhost -U postgres -d it4065c
- ***Password:*** *Pa$$w0rd123!*
- ***✓Success looks like:***
  it4065c=#

If you cannot connect, stop here and resolve before continuing.

```
kofi@18ntiikxil02:~/IT4065C-Labs$ psql -h localhost -U postgres -d it4065c
Password for user postgres:
psql (14.20 (Ubuntu 14.20-0ubuntu0.22.04.1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

it4065c=#
```

Step 2: Ensure Raw Data Exists

Step 2.1: Check for raw tables (inside psql)

- Run:

  \dt raw.*

If you see tables (customers, orders, products, order_items):

```
it4065c=# \dt raw.*
Did not find any relation named "raw.*".
it4065c=#
```

✅ Proceed to Step 3

- If you do NOT see any tables:
- Exit psql

  \q
- ***Make sure you are in the repo folder (cd ~/IT4065C-Labs)***
- Run the seed file once:

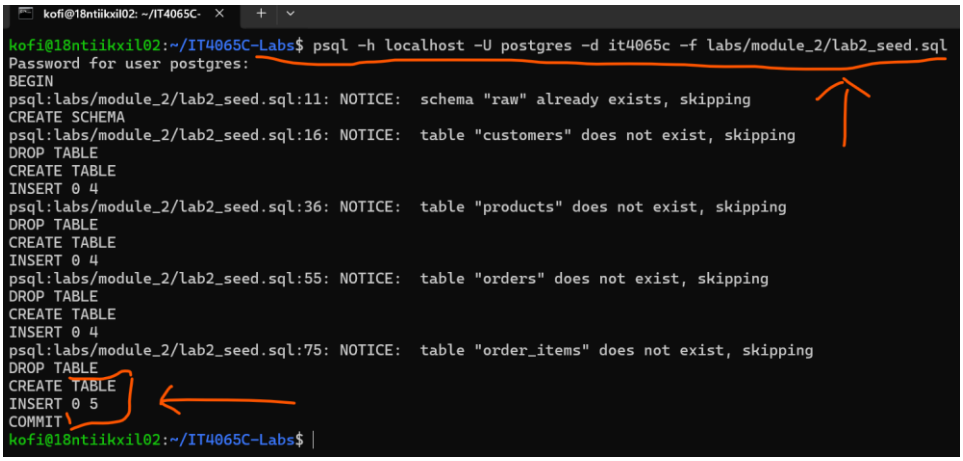
  psql -h localhost -U postgres -d it4065c -f labs/module_2/lab2_seed.sql
- Reconnect and re-check:
  - Run the following (one after the other):

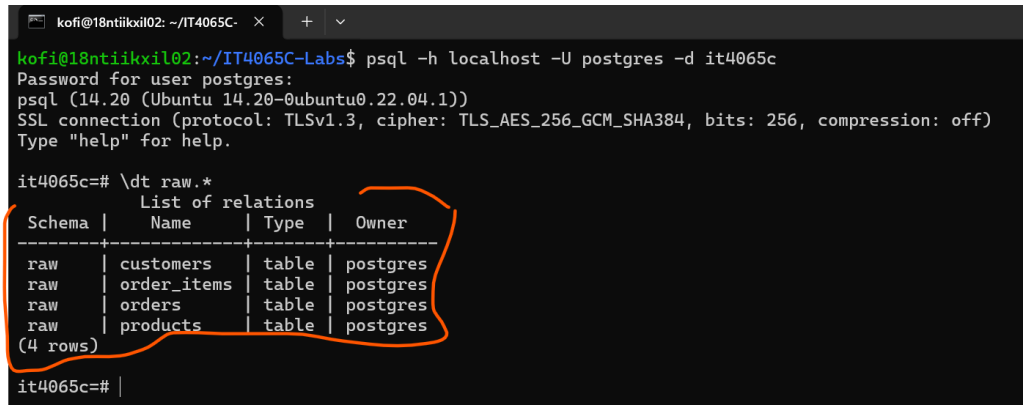
    psql -h localhost -U postgres -d it4065c

    \dt raw.*

This step ensures everyone works with the same clean dataset, reducing confusion.

Screenshot 1: Take a screenshot showing the raw tables listed.





Step 3: Explore Raw Tables (Think Like a Data Steward)
You will explore at least two raw tables.
Step 3.1: Inspect table structure (inside psql)

- Run the following (one after the other)**:**
  \d raw.customers
  \d raw.orders

This tells you:

- Column names
- Data types
- Which fields *might* be sensitive

```
it4065c=# \d raw.customers
                                        Table "raw.customers"
    Column    |            Type             | Collation | Nullable |                    Default
--------------+-----------------------------+-----------+----------+-----------------------------------------------
 customer_id  | integer                     |           | not null | nextval('raw.customers_customer_id_seq'::regclass)
 first_name   | text                        |           | not null |
 last_name    | text                        |           | not null |
 email        | text                        |           | not null |
 phone_number | text                        |           |          |
 created_at   | timestamp without time zone |           |          | now()
Indexes:
    "customers_pkey" PRIMARY KEY, btree (customer_id)

it4065c=# \d raw.orders
                                         Table "raw.orders"
     Column     |            Type             | Collation | Nullable |                   Default
----------------+-----------------------------+-----------+----------+-------------------------------------------
 order_id       | integer                     |           | not null | nextval('raw.orders_order_id_seq'::regclass)
 customer_id    | integer                     |           | not null |
 order_date     | timestamp without time zone |           |          | now()
 order_status   | text                        |           | not null |
 total_amount   | numeric(10,2)               |           | not null |
 payment_method | text                        |           | not null |
Indexes:
    "orders_pkey" PRIMARY KEY, btree (order_id)

it4065c=#
```

Step 3.2: Preview sample data (safe amount)

- Run the following SQL command (one after the other)
  SELECT * FROM raw.customers LIMIT 10;
  SELECT * FROM raw.orders LIMIT 10;

⚠️ Do not dump entire tables.

```
it4065c=# SELECT * FROM raw.customers LIMIT 10;
 customer_id | first_name | last_name |          email          | phone_number |         created_at
-------------+------------+-----------+-------------------------+--------------+----------------------------
           1 | Alice      | Johnson   | alice.johnson@email.com | 513-555-1023 | 2026-01-19 23:48:26.424667
           2 | Brian      | Smith     | brian.smith@email.com   |              | 2026-01-19 23:48:26.424667
           3 | Carla      | Nguyen    | carla.nguyen@email.com  | 513-555-7781 | 2026-01-19 23:48:26.424667
           4 | David      | Brown     | david.brown@email.com   |              | 2026-01-19 23:48:26.424667
(4 rows)

it4065c=# SELECT * FROM raw.orders LIMIT 10;
 order_id | customer_id |         order_date         | order_status | total_amount | payment_method
----------+-------------+----------------------------+--------------+--------------+----------------
        1 |           1 | 2026-01-19 23:48:26.424667 | Completed    |        79.98 | Credit Card
        2 |           2 | 2026-01-19 23:48:26.424667 | Completed    |        49.99 | Gift Card
        3 |           3 | 2026-01-19 23:48:26.424667 | Cancelled    |        19.95 | Store Credit
        4 |           1 | 2026-01-19 23:48:26.424667 | Completed    |        34.98 | Credit Card
(4 rows)

it4065c=#
```

Step 4: Identify Sensitive Data
As you explore tables, ask:

- Does this identify a person?
- Does this involve money or payment?
- Would misuse of this data cause harm?

Classification Levels You Will Use

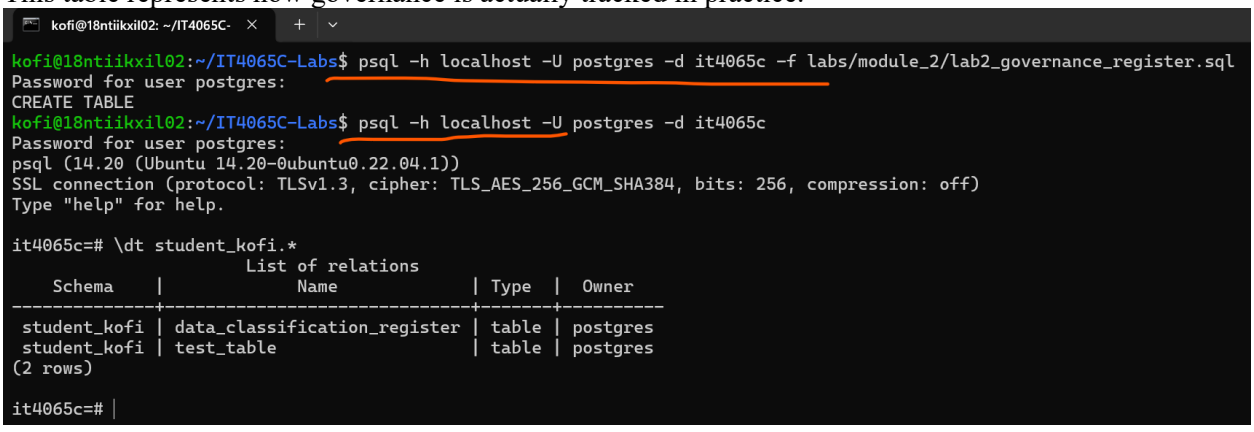
Level	Meaning
Public	Safe to share broadly
Internal	Business-use only
Sensitive	PII or financial data
Restricted	High-risk personal or financial data

There is no single "correct" answer; your reasoning matters.

Step 5: Create a Governance Register (Your Workspace)
You will document your decisions in your own schema.
Step 5.1: Initialize the Governance Register

- Exit psql
  \q
- *Make sure you are in the repo folder (cd ~/IT4065C-Labs).*
- Run the provided register script:
  psql -h localhost -U postgres -d it4065c -f labs/module_2/lab2_governance_register.sql
- Then connect to psql and verify
  - Run the following commands (One after the other):
    psql -h localhost -U postgres -d it4065c
    \dt student_kofi.*

This table represents how governance is actually tracked in practice.



Step 6: Populate the Register (Main Task)

1. Exit psql if you are inside it
   \q
2. *Make sure you are in the repo folder (cd ~/IT4065C-Labs).*
3. Open the provided insert template
   nano labs/module_2/lab2_insert_templates.sql
4. **Edit ONLY the placeholder values**
   - Do **not** change the SQL structure
   - Replace values such as:
     - table name
     - column name
     - classification
     - rationale
     - owner role
     - retention type

   The file already contains **one INSERT block**.
   You only need to complete that one block.
5. **Save and exit**
   - Ctrl + O → Enter
   - Ctrl + X
6. **Run the completed script**
   psql -h localhost -U postgres -d it4065c -f labs/module_2/lab2_insert_templates.sql

---

Quality Guidance (Important)

✓ Your **rationale must be specific**

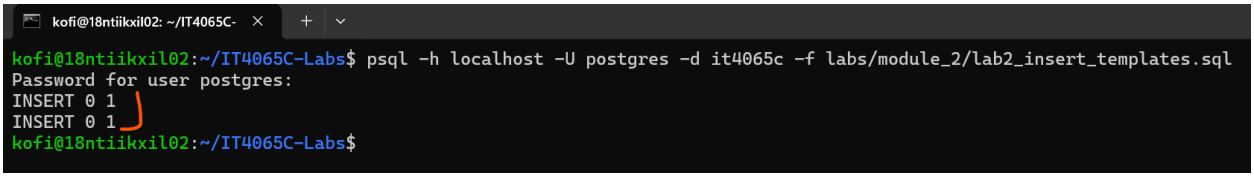
❌ Avoid vague explanations like *"because it is PII"*

Good rationale example: "This field directly identifies an individual customer and could be misused if exposed."

Success Check

After running the script, you should have:

- **At least one row** in student_kofi.data_classification_register
- A classification you can clearly explain in plain language



Step 7: Verify Your Work

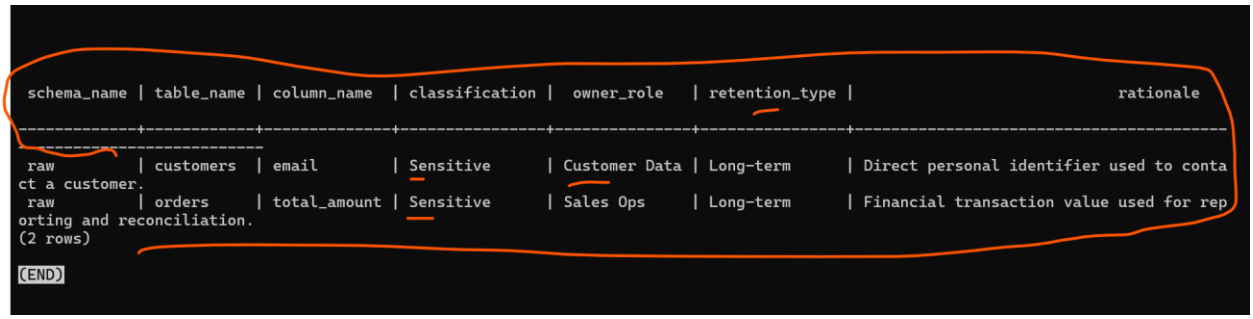
- Make sure you are in the repo folder (cd ~/IT4065C-Labs):

psql -h localhost -U postgres -d it4065c -f labs/module_2/lab2_verify_register.sql

Review the output:

- Do you see entries for both customers and orders?
- Do your counts match your intended number of classifications?
- Do the classifications make sense when viewed together?

Screenshot 2: Take a screenshot of the verification output showing entries/counts.



```
 schema_name | table_name | column_name  | classification |  owner_role   | retention_type |                                rationale
-------------+------------+--------------+----------------+---------------+----------------+-------------------------------------------
 raw         | customers  | email        | Sensitive      | Customer Data | Long-term      | Direct personal identifier used to conta
ct a customer.
 raw         | orders     | total_amount | Sensitive      | Sales Ops     | Long-term      | Financial transaction value used for rep
orting and reconciliation.
(2 rows)

(END)
```

Success check: You should see at least one classified column from customers and one from orders.
Note: If you see (END) at the bottom of the screen, press **q** to return to the prompt.

Step 8: Reflection (Short)

In **3–6 sentences**, answer:

1. Which columns were easiest to classify, and why?
2. Which columns were uncertain?
3. What additional business context would help you decide?
4. How does separating raw data from governed data help with accountability?

Deliverables

Submit:

- Screenshot 1: raw tables listed (\dt raw.*)
- Screenshot 2: verification output (shows your register entries/counts)
- Short reflection (3–6 sentences)

Final Note
If something feels confusing, that is expected. Data governance is judgment-based, not formula-based. Your goal is to explain your reasoning, not guess the instructor's answer.