NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE | NTU Academy for Professional and Continuing Education

(SCTP) Advanced Professional Certificate

Data Science and AI

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

C2.2 Coaching session

22 Nov 2025

Agenda

- 2.4 Web Scraping (Continued)
- Sample Module 2 Project
- Project excellence criteria 15m
- DBT tool overview
- Technical Setup for Lesson 2.5-2.7 (DBT, Meltano, Dagster)

Sample Module 2 Project

Module 2 Project Presentation & Deck

Presentation

- 10 minutes presentation, 5 minutes Q&A
- Focus on both technical design and business impact / value

Deck

- 10-15 pages
- Presentation Flow Sample:
    - Setup (Introduction/Problem)
    - Methodology (How you approached it)
    - Analysis (What you found)
    - Implementation/Recommendations (What to do about it)
    - Technical Details (How it works)

Example Presentation Outline

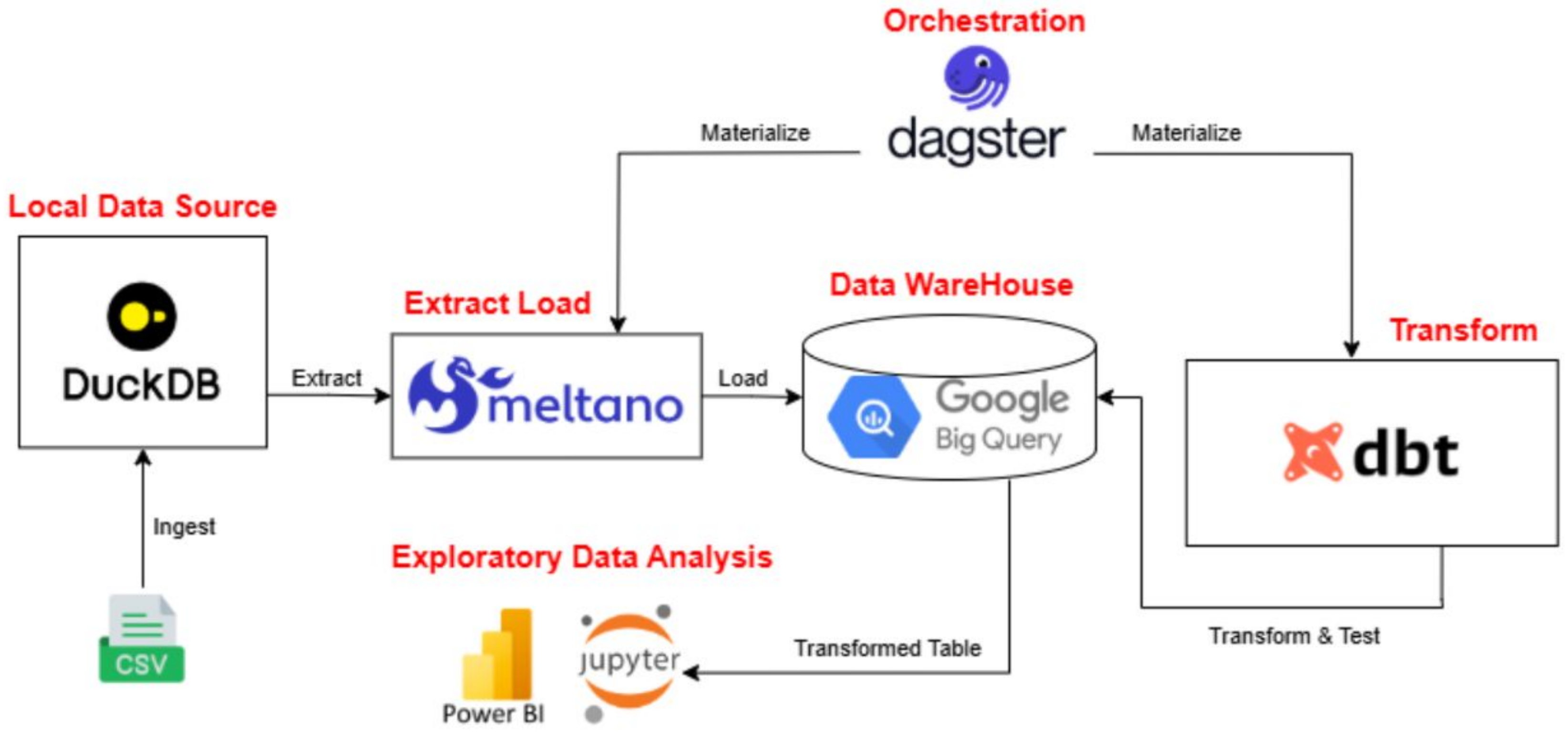
Olist E-commerce Analysis Outline:

1. Executive Summary
    - High-level analysis purpose and focus
2. Introduction & Context
    - Problem statement and scope
    - Performance insights methodology
3. Methodology & Data
    - Dataset description (Olist e-commerce data)
    - Approach and assumptions
    - Data description and exclusions
4. Results & Insights
    - Key metrics dashboard
    - E-commerce sales insights
    - Analysis and findings
5. Strategic Recommendations
    - Actionable recommendations based on findings
6. Conclusion
    - Summary and strategic implications

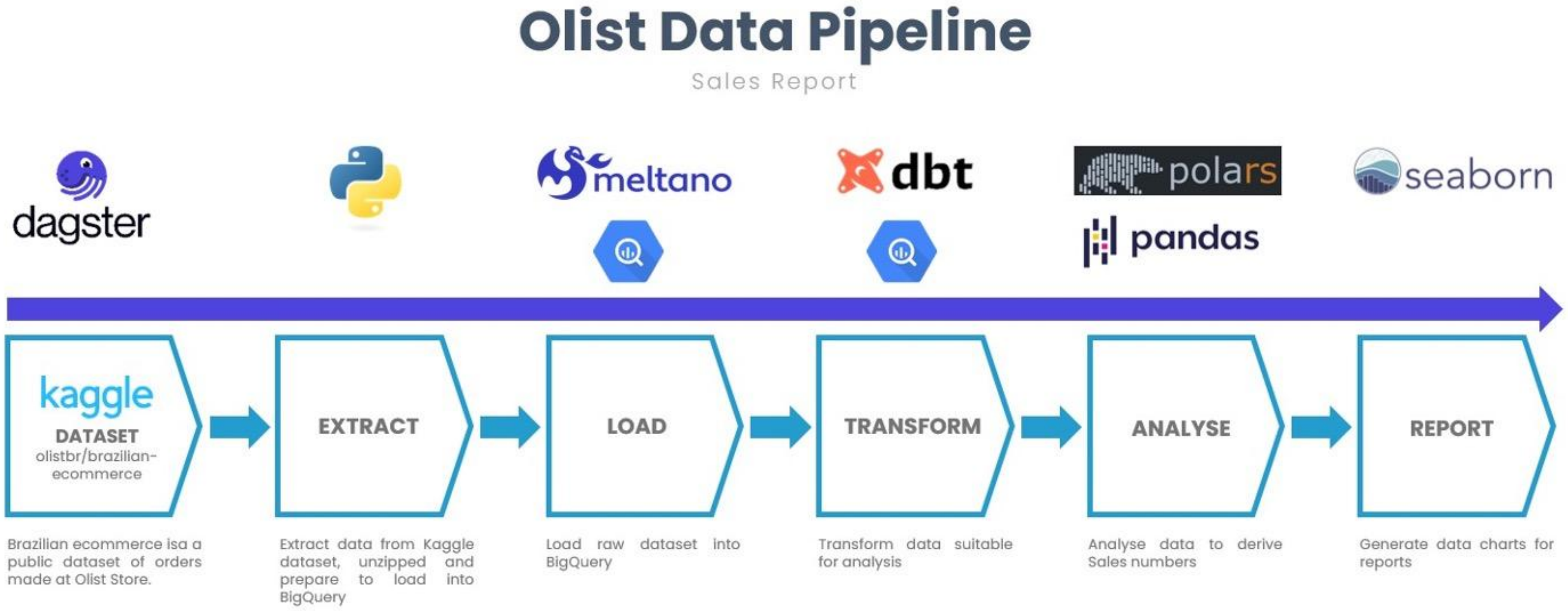
Citi Bike Analysis Outline :

1. Introduction & Setup
    - Dataset description (Citi Bike Trip Histories)
    - Project goals (automated ELT pipeline)
    - Architecture overview
2. Exploratory Data Analysis (EDA)
    - Usage trends analysis
    - Revenue and trip composition
    - Station popularity analysis
    - Anomaly detection
    - Potential failure rate analysis
3. Technical Implementation
    - Database schema design (star schema)
    - Data ingestion process (Meltano orchestration)
    - Key implementation logic for target tables
    - Dagster orchestration and assets

Sample Implementation 1



Sample Implementation 2


Olist Data Pipeline
Sales Report
dagster
meltano
dbt
polars
pandas
seaborn
kaggle
DATASET
olistbr/brazilian-ecommerce
EXTRACT
LOAD
TRANSFORM
ANALYSE
REPORT
Brazilian ecommerce isa a public dataset of orders made at Olist Store.
Extract data from Kaggle dataset, unzipped and prepare to load into BigQuery
Load raw dataset into BigQuery
Transform data suitable for analysis
Analyse data to derive Sales numbers
Generate data charts for reports

Excellence Criteria

Module 2 Final Project - Excellence Criteria 1/2

1

Data Ingestion

- Successfully ingested data to the data warehouse.
- Justified ingestion decisions.
- Compared various methods with pros and cons.

2

Data Warehouse Design

- Designed a star schema.
- Created most dimension and fact tables.
- Provided clear justifications for design decisions.

3

ELT Pipeline

- Implemented ELT pipelines with data cleaning and validation.
- Incorporated derived columns and business logic (e.g., fact tables).
- Justified the design process with clear reasoning.

4

Data Quality Testing

- Utilized testing tools extensively.
- Achieved good coverage in test cases.
- Implemented additional test cases beyond those taught in class.

Module 2 Final Project - Excellence Criteria 2/2

5

Data Analysis with Python

- Attempted data analysis with key metrics.
- Provided justifications for metric importance.
- Linked analysis to data pipeline design decisions.

6

Pipeline Orchestration (Optional)

- Automated entire pipeline using an orchestration framework.
- Provided clear justifications for framework choice.
- Listed pros and cons compared to other frameworks.

7

Documentation

- Documented code, data lineage, and pipeline architecture (e.g., DRAW.IO, EXCALIDRAW).
- Prepared comprehensive report summarizing technical approach, findings, and insights.
- Included relevant tables/charts/graphs.
- Explained tool choices and schema design justifications.

8

Executive Stakeholder Presentation

- Delivered a highly polished and persuasive presentation for a mixed executive audience.
- Clearly articulated business value and actionable insights ("the so what").
- Confidently justified technical architecture and tool choices.
- Addressed scalability, maintainability, cost, and direct business impact.
- Professionally handled questions.

DBT Tool Overview

DBT Introduction - Drew Banin - DBT Co-Founder

What is dbt

"dbt (data build tool) is an open-source command-line tool used in data engineering to transform data directly within a data warehouse using SQL. It focuses on the "T" (transform) in the ELT (Extract, Load, Transform) process, enabling data teams to build and manage data models, apply testing and documentation, and version-control their transformations to create clean, analysis-ready data."

– *Google AI Overview*

Why Learn DBT?

Reusability

Build features and scoring logic **once** and access them everywhere

Reliability

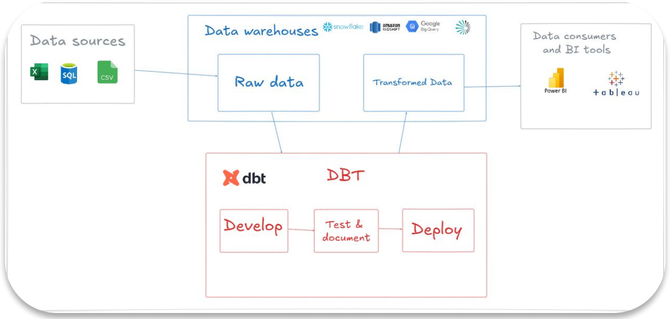
Test, version-control, and document your data transformations

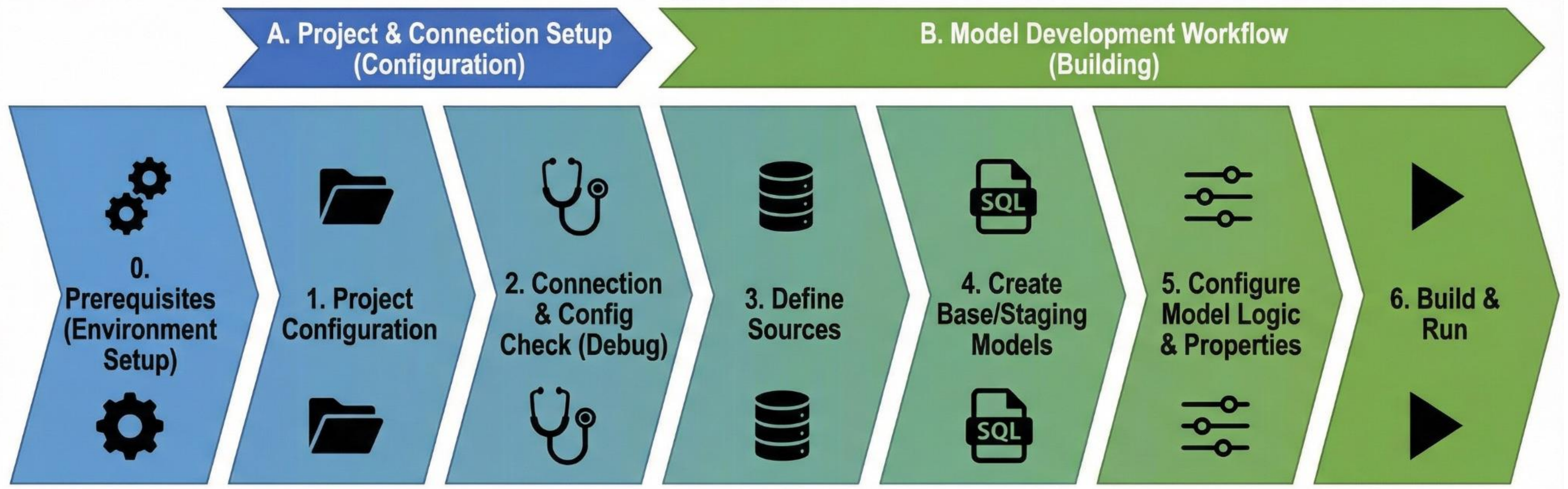
Efficiency

Transform data once and reuse across Python, dashboards, and ML models

Collaboration

Stakeholders can interact with model outputs directly without code access




A. Project & Connection Setup (Configuration)
B. Model Development Workflow (Building)
0. Prerequisites (Environment Setup)
1. Project Configuration
2. Connection & Config Check (Debug)
3. Define Sources
4. Create Base/Staging Models
5. Configure Model Logic & Properties
6. Build & Run
SQL

Config File: ~/.dbt/profiles.yml

Explanation: Stores credentials (e.g., BigQuery key), defines targets (dev/prod). Lives OUTSIDE repo.

Config File: dbt_project.yml

Explanation: Defines project name, profile reference, model paths, and global settings.

Command: dbt debug
Explanation: Verifies warehouse connection (profiles.yml) and validates project configuration. Ensures system is ready.

Config File: sources.yml (or similar .yml)

Explanation: Documents raw tables, schemas, and freshness checks.

Config File: .sql files & schema.yml

Explanation: Writes initial SQL transformations, defines basic properties and column tests in YAML.

Config File: .sql files & schema.yml

Explanation: Refines SQL, applies specific configs (materialization, tags) via {{ config() }} jinja or YAML

Command: dbt run

Explanation: Executes compiled SQL against DW to create tables or views.

DBT Project & Connection Configuration

dbt_project.yml

Defines project structure, model paths, and default configurations

```
name:           'my_project'
version:        '1.0.0'
model-paths:    ["models"]
target-path:    "target"
```

profiles.yml

Contains connection settings and environment configurations

```
my_project:
  target:     de
  outputs:    v
    dev:
      type:     snowflake
      account:      abc123
```

packages.yml

Lists external dbt packages and their versions

```
packages:
- package:        dbt-labs/dbt_utils
    version:      1.0.0
- local:        ../my_local_package
```

DBT Schema & Model Configuration

schema.yml

Defines sources, models, columns, and tests in YAML format

```yaml
version: 2
sources:
  - name: raw_data
    tables:
      - name: customers
        description: Customer data
        columns:
          - name: customer_id
            tests:
              - unique
              - not_null
```

model.sql

SQL SELECT statement that defines a transformation

```sql
{{ config(materialized='table') }}
SELECT
    customer_id,
    first_name,
    last_name,
    email
FROM {{ ref('stg_customers') }}
```

DBT Schema & Model Configuration

models/

Contains SQL model files organized by layer and purpose

```
models/
staging/
    stg_customers.sql
    stg_orders.sql
intermediate/
    int_customer_orders.sql
marts/
    dim_customers.sql
    fct_orders.sql
```

Technical Setup for Lesson 2.5-2.7 (DBT, Meltano, Dagster)

Lesson 2.5 Prerequisite

- Conda environment: `elt`
- GCP CLI: make sure GCP CLI is authenticated and able to run command
- GCP Project ID

Lesson 2.6 Prerequisite

- Conda environment: `elt, dagster`
- GCP service key file: place the service key file at a location and get ready the path to the service key file (WSL users copy path from VSCode, **Do Not** use explorer)
- GCP project ID
- Github token

Lesson 2.7 Prerequisite

- Conda environment: `elt, dagster`
- Github token
- File path to the meltano and dbt project folder in lesson 2.6

Some dbt Commands:

- `dbt debug` - Test connection to data warehouse (e.g. BigQuery)
- `dbt run` - Run dbt models for the project
- `dbt test` - Executes test defined on the project
- `dbt snapshot` - Execute snapshot job
- `dbt docs` - generate document
- `dbt init` - Initialize a new dbt project

- *https://docs.getdbt.com/docs/core/connect-data-platform/about-core-connections*
- *https://docs.getdbt.com/reference/dbt-commands*

dbt Profiles Overview

- `profiles.yml` is the file that contains your data warehouse configuration details

By default, dbt looks for `profiles.yml` in:

- The dbt project folder where you run your dbt commands
- Otherwise (Mac), `/Users/<USER>/.dbt/profiles.yml`
- Otherwise (Linux/WSL), `/home/<USER>/.dbt/profiles.yml`
- Command `dbt init` will set profiles at home folder shown above.
- A single `profiles.yml` can define **multiple** dbt projects

- *https://docs.getdbt.com/docs/core/connect-data-platform/profiles.yml*

dbt Connection with BigQuery

- Using `oauth` - This is easiest and straightforward method if your GCP is setup correctly.
- Using `service key file` - This is best for deployment. Please note that for private repository in Github, you can place your key file inside the repository.
- For public repository - you need to place them in secrets manager on the deployment platform you are using.
- Github will remove your key if you commit your public repository with your service key file.

Best Practices (Do’s)

- **Always** run dbt command using correct conda environment.
- **Always** run `dbt debug` before `dbt run`
- **Always** run dbt command on the dbt project folder
- Optionally, run `dbt clean` first to clear any previous state especially if you have many failed attempts.
- Ensure the **profile name** in `profiles.yml` matches the one in `dbt_project.yml`

Pitfalls to Avoid (Don’ts)

❌ Do **not** create a new project inside the folder of an existing dbt project

❌ Do **not** run `dbt init` in an already-initialized project folder

❌ Do **not** reuse the same dbt folder name for two different projects

Troubleshooting Checklist 1

When `dbt debug` fails:

1. Are you in the correct Conda/python environment?
   - `conda activate elt`

2. Are you inside your dbt project folder?
   - e.g. `cd liqour_sales`

3. Is `profiles.yml` in your project folder or home location?

4. Is `dbt_project.yml` in the folder where you ran dbt debug?

5. Do the profile names match in both `profiles.yml` and `dbt_project.yml`?

Troubleshooting Checklist 2

When `dbt run` fails:

- Often indicates missing dependencies; check if you need to run first:
  - `dbt snapshot`
  - `dbt deps`
- Check your data source
- Check your SQL construct (Can try SQL command inside Bigquery first)