dbt Development Process

This document outlines the dbt development workflow used in this repository. It describes how to work with the **silver layer** (`consolidation/`) and the **gold layer** (`dataconsumption/`), organize sources, write models and tests, use contracts, and run dbt commands effectively.

Project Structure

```
cai_bi/
├── models/
│   ├── consolidation/       # Silver layer: staging/cleaned models
│   │   ├── model.sql        # Each model file
│   │   └── model.yml        # Matching metadata and tests
│   └── dataconsumption/     # Gold layer: business/reporting models
├── seeds/                   # Static reference data (CSV)
├── macros/                  # Reusable Jinja macros
├── tests/                   # Custom test SQLs
├── profiles/                # dbt profile config (used at runtime)
├── dbt_project.yml          # Core project config
└── logs/, snapshots/, target/ (support files and outputs)
```

Sources

Define raw Snowflake tables as sources in a YAML file (typically under `models/sources/` or `models/consolidation/`):

```yaml
version: 2

sources:
  - name: raw_data
    schema: RAW
    database: MY_DB
    tables:
      - name: employee_raw
        description: "Raw employee data from HR system"
```

Use in models:

```
SELECT * FROM {{ source('raw_data', 'employee_raw') }}
```

Models and YAML Metadata

Each model `.sql` file should have a matching `.yml` file (same name) defining:

- Model and column descriptions
- Data quality tests
- Contracts (optional)

Example: `employee.yml`

```
version: 2

models:
  - name: employee
    description: "Cleaned and joined employee dimension"
    config:
      contract:
        enforced: true  # Enforces column names and types
    columns:
      - name: employee_id
        description: "Primary key"
        tests:
          - not_null
          - unique
      - name: email
        description: "Work email"
        tests:
          - not_null
          - dbt_expectations.expect_column_values_to_match_regex:
              regex: '^.+@.+\..+$'
```

Use `ref()` to chain models:

```
SELECT * FROM {{ ref('project') }}
```

Testing

Tests defined in `.yml` files validate model data quality.

Run All Tests

```
dbt test
```

Run Tests for a Specific Model

```
dbt test --select employee
```

Run Tests for a Specific Column

```
dbt test --select employee.email
```

▶ Building Models

Run All Models

```
dbt run
```

Run a Specific Model

```
dbt run --select employee
```

Run a Folder or Graph Segment

```
dbt run --select consolidation/
dbt run --select +project  # project and its dependencies
dbt run --select timesheet_entry+  # timesheet_entry and all downstream models
```

🔎 Viewing Compiled SQL Queries

When you run dbt, it compiles your models by rendering all Jinja, macros, and references into raw SQL.

🗀 Inspect Compiled Files

Compiled SQL files are saved under:

```
target/compiled/
```

For example:

```
target/compiled/cai_bi/models/consolidation/employee.sql
```

You can open these files in your editor to inspect exactly what SQL was executed.

Print SQL to Terminal with `--debug`

Add the `--debug` flag to any `dbt run` or `dbt build` command to print compiled queries directly to the terminal:

```
dbt run --select employee --debug
```

```
dbt build --select dataconsumption --debug
```

This is useful when you want to:

- See the actual SQL before it's executed
- Understand how your `ref()` and Jinja logic are being resolved
- Troubleshoot issues without opening the compiled files

`dbt build`: All-in-One

```
dbt build
```

This runs:

- `dbt run`
- `dbt test`
- `dbt seed`
- `dbt snapshot` (if applicable)

You can also build specific models:

```
dbt build --select expenseitem
```

Use `--debug` with `build` to print compiled SQL:

```
dbt build --select dataconsumption --debug
```

Seeds

CSV files under `seeds/` can be loaded with:

```
dbt seed
```

Use them in models like:

```
SELECT * FROM {{ ref('country_lookup') }}
```

Documentation

Generate and view model documentation:

```
dbt docs generate
dbt docs serve
```

Includes:

- Model and column descriptions
- Source documentation
- Data lineage graphs
- Test coverage

Maintenance Commands

Clean up build artifacts:

```
dbt clean
```

Validate your dbt setup and connection:

```
dbt debug
```

References

- dbt Core Docs
- dbt Snowflake Profile
- dbt Contracts
- dbt Tests
- dbt Documentation
- installation_and_setup