dbt Modular Project Structure

Layered dbt architecture with staging, intermediate, and marts. Clear boundaries, explicit dependencies, and team-friendly organization.

#dbt #sql #data #analytics #warehouse #medallion #layers

PNG PDF Copy Prompt

Project Directory

```
dbt_project/
  models/
    staging/  1:1 with sources
      stripe/
        _stripe__sources.yml
        _stripe__models.yml
        stg_stripe__payments.sql
        stg_stripe__customers.sql
      shopify/
        _shopify__sources.yml
        _shopify__models.yml
        stg_shopify__orders.sql
        stg_shopify__products.sql
    intermediate/  Business logic
      finance/
        _int_finance__models.yml
        int_payments_pivoted.sql
    marts/  Business entiti…
      core/
        _core__models.yml
        dim_customers.sql
        fct_orders.sql
      finance/
        _finance__models.yml
        fct_revenue.sql
      marketing/
        _marketing__models.yml
        dim_campaigns.sql
  seeds/
    country_codes.csv
  tests/
    assert_positive_revenue.sql
  macros/
    generate_schema_name.sql
    cents_to_dollars.sql
  snapshots/
    customers_snapshot.sql
  dbt_project.yml
  packages.yml  dbt packages
  .gitignore
  README.md
```

Why This Structure?

The staging → intermediate → marts pattern creates clear data lineage. Staging models are 1:1 with sources (rename, cast, basic cleaning). Intermediate handles complex joins and business logic. Marts are the final dimensional models consumed by BI tools.

Key Directories

models/staging/ - One subfolder per source system (stripe/, shopify/)

models/intermediate/ - Complex transformations, not exposed to end users

models/marts/ - Final dim_ and fct_ tables for BI consumption

_*__models.yml - Per-folder schema files for docs and tests

Getting Started

1. `dbt init my_project && cd my_project`
2. Create `staging/{source}/` folder for each data source
3. Use `stg_` prefix for staging, `int_` for intermediate, `dim_` / `fct_` for marts
4. `dbt run --select staging` to build layer by layer

Naming Conventions

stg_stripe__payments - Staging: stg_{source}__{table}

int_payments_pivoted - Intermediate: int_{description}

dim_customers - Dimension: dim_{entity}

fct_orders - Fact: fct_{event/transaction}

Mart Model

```sql
-- models/marts/core/fct_orders.sql
with orders as (
    select * from {{ ref('stg_shopify__orders') }}
),

payments as (
    select * from {{ ref('int_payments_pivoted') }}
)

select
    orders.order_id,
    orders.customer_id,
    orders.order_date,
    payments.total_amount
from orders
left join payments using (order_id)
```

When To Use This

- Teams with 20-100+ models
- Multiple data sources to integrate
- Need clear ownership by domain (finance, marketing)
- BI tools consuming final marts layer

Best Practices

- Staging models should only SELECT from sources
- Never skip layers—marts should ref staging or intermediate, not sources
- Use `__{source}__` double underscore naming convention
- One `_models.yml` per folder for maintainability
- Materialize staging as views, marts as tables