**What is a dbt Contract?**

A **dbt contract** is a feature that allows you to **enforce schema definitions** for a dbt model.  
When you enable contracts, dbt validates that the actual database table or view matches the schema you’ve defined in your model’s .yml file.

This ensures that:

* Columns exist as expected
* Column data types match
* Extra or missing columns cause errors

In simple terminology, it’s a **schema enforcement mechanism** that prevents accidental drift between your code and your warehouse tables.

**Advantages of dbt Contracts**

1. **Data Reliability**  
   Ensures downstream teams always get the expected schema (no surprise column changes).
2. **Early Error Detection**  
   Build will fail immediately if schema mismatches occur.
3. **Governance & Consistency**  
   Standardizes column naming and types across models.
4. **Stronger CI/CD Integration**  
   Contracts act like "tests" for schema, making deployments safer.
5. **Documentation Accuracy**  
   Your .yml files (source of truth) always align with the actual database schem

## Disadvantages of dbt Contracts

1. **Rigidity**  
   Small schema changes require updating contracts, which may slow development.
2. **Backward Compatibility Issues**  
   If downstream systems depend on old schemas, enforcing contracts can break builds until everyone migrates.
3. **Overhead for Prototyping**  
   Extra setup may feel heavy for fast iteration or ad-hoc analysis.
4. **Not All Databases Supported Equally**  
   Some warehouses (e.g., BigQuery, Snowflake, Redshift) handle contracts well, but edge cases may exist with data type enforcement.

## How to Implement dbt Contracts:

1. Define the model schema in .yml:
2. Enable the contract in your model’s config:
3. Run the model:

If the database schema doesn’t match the contract, dbt will fail the build.