* The dbt test command is used to execute the tests you defined for your project. This ensures that all of your tests are passing. The dbt run command is used to run the models you defined in your project, while the dbt build command is used to build and test resources such as models, seeds, snapshots, and tests. The dbt deploy command is not a valid command in dbt.
* Profile.ym; file generally lives outside of your dbt project to avoid sensitive credentials being checked in to version control, but profiles.yml can be safely checked in when [using environment variables](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#advanced-using-environment-variables) to load sensitive credentials.
* ***If your user is unable to be granted the privilege to create schemas, your dbt runs should instead target an existing schema that your user has permission to create relations within.***
* ***dbt makes it easy to maintain separate production and development environments through the use of***[***targets***](https://docs.getdbt.com/reference/dbt-jinja-functions/target)***within a***[***profile***](https://docs.getdbt.com/docs/core/connect-data-platform/profiles.yml)*.*
* ***Once you are confident in your changes, you can deploy the code to production, by running your dbt project with a prod target.***
* ***! We recommend using different schemas within one database to separate your environments. This is the easiest to set up and is the most cost-effective solution in a modern cloud-based data stack.***
* ***If you have multiple dbt users writing code, it often makes sense for each user to have their own development environment. A pattern we've found useful is to set your dev target schema to be dbt\_<username>. User credentials should also differ across targets so that each dbt user is using their own data warehouse user.***
* Note that there’s no need to create your target schema beforehand – dbt will check if the schema already exists when it runs, and create it if it doesn’t.
* If using this method, the --profiles-dir option needs to be provided every time you run a dbt command.
* **Use the DBT\_PROFILES\_DIR environment variable to change the default location**[**​**](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#2-use-the-dbt_profiles_dir-environment-variable-to-change-the-default-location)

Specifying this environment variable overrides the directory that dbt looks for your profiles.yml file in. You can specify this by runnin

Incremental model

* *For more complex incremental models that make use of Common Table Expressions (CTEs), you should consider the impact of the position of the is\_incremental() macro on query performance.* ***In some warehouses, filtering your records early can vastly improve the run time of your query!***
* Either ensure that each column has no nulls (for example with coalesce(COLUMN\_NAME, 'VALUE\_IF\_NULL')), or define a single-column [surrogate key](https://docs.getdbt.com/terms/surrogate-key) (for example with [dbt\_utils.generate\_surrogate\_key](https://github.com/dbt-labs/dbt-utils" \l "generate_surrogate_key-source" \t "_blank)).
* When you pass a list of columns in unique key, please ensure that each column does not contain any nulls, or the incremental model run may fail.
* If you provide the --full-refresh flag to dbt run, dbt will treat incremental models as [table](https://docs.getdbt.com/terms/table) models. This is useful when

The schema of an incremental model changes and you need to recreate it.

You want to reprocess the entirety of the incremental model because of new logic in the model code.

* We can define the incremental strategy in both of the ways

1. Defining it in dbt\_project.yml
2. Defining in the models sections where incremental strategy is defined

* **Incremental predicates** in dbt are an advanced feature that allows you to further optimize the processing of incremental models by limiting the data scan of the existing table. This can be particularly useful when dealing with large volumes of data.

To use incremental predicates, you need to configure the **incremental\_predicates** option in your model configuration. This option accepts a list of valid SQL expressions that define the conditions for limiting the scan. The expressions can reference the columns in the target table.

**incremental\_predicates** = **["date\_column > dateadd(day, -7, current\_date)"]** )

* **FULL REFRESH:** IF WE **ENABLE: TRUE** THE FULL REFRESH IN CONFIG THEN IT WILL ALWAYS RUN IT EVEN WITHOUT ASKING FOR IT IN CLI FLAG AND DBT –FULL-REFERESH IS INVOKED. AND IF WE MARK IT AS **FALSE** THEN EVEN INVOKING IT FROM CLI FLAG IT WILL NOT ACCEPT IT AND WILL NOT REFERESH IT.

Source Freshness:

* It is a column name (or expression) that returns a timestamp indicating freshness. If using a date field, you may have to cast it to a timestamp: **loaded\_at\_field:** **"completed\_date::timestamp"** If using a non-UTC timestamp, cast it to UTC first: **loaded\_at\_field:** **"convert\_timezone('UTC', 'Australia/Sydney', enter\_column(created\_at\_local))"**
* When dbt source freshness completes, a [JSON](https://docs.getdbt.com/terms/json) file containing information about the freshness of your sources will be saved to **target/sources.json**
* **Check freshness for all sources and tables:**

dbt source snapshot-freshness

* **Check freshness for specific sources or tables:**

dbt source snapshot-freshness --select source:my\_source\_name

dbt source snapshot-freshness --select table:my\_source\_name.my\_table\_name

* **Exclude specific sources or tables from freshness check:**

dbt source snapshot-freshness --exclude source:my\_excluded\_source\_name

* **To override the destination for this sources.json file, use the -o (or --output) flag:**  
  # Output source freshness info to a different path  
  $ **dbt source freshness** **--output target/source\_freshness.json**

**Which of the following dbt flags can be used with the dbt test command? (Select all that apply)**

Choose only ONE best answer.

**A**

--fail-fast

**B**

--full-refresh

**C**

--store-failures

**D**

--partial-parse

`