dbt Analytics Engineering Certification Exam Study Guide

dbt Labs

How to use this study guide

This is the official study guide for the **dbt Analytics Engineering Certification Exam** from the team at dbt Labs. While the guide suggests a sequence of courses and reading material, we recommend using it to supplement (rather than substitute) real-world use and experience with dbt.

The **exam overview** will provide high-level details on the format of the exam. We recommend being mindful of the number of questions and time constraints. The exam supports version 1.7

The **topic outline** will provide a clear list of the topics that are assessed on the exam. dbt subject matter experts used this topic outline to write and meticulously review all of the exam items that you will find on the exam.

The **sample exam questions** provide examples of some of the formats to expect for the exam. The types of questions you can expect on the exam include

- Multiple-choice
- Fill-in-the-blank
- Matching
- Hotspot
- Build list
- Discrete Option Multiple Choice (DOMC)

The **learning path** will walk you through a suggested series of courses, readings, and documentation. We will also provide some guidance for the types of experience to build while using dbt on a real-world project.

Finally, the **additional resources** section will provide additional places to continue your learning.

Exam Overview

The **dbt Analytics Engineering Certification Exam** is designed to evaluate your ability to

- build, test, and maintain models to make data accessible to others
- use dbt to apply engineering principles to analytics infrastructure

We recommend that you have at least SQL proficiency and have had 6+ months of experience working in dbt (Core or Cloud) before attempting the exam.

Logistics

- Duration: 2 Hours
- Format & Registration: online proctored
- Length: 65 questions
- Supported Version: dbt core 1.7
- Passing Score: 65% or higher. You will know your score immediately after completion of the exam.
- Price: $200*
- Language: Only English at this time
- Certification Expiration: The certification expires 2 years after the date awarded.
- Supported browsers: Caveon Web browsers

*The price applies only to the online proctored exam; there is no additional fee for Coalesce premium attendees. Discounts are available for dbt Labs SI Partners.

Scoring

The exam is scored on a point basis, with 1 point for each correct answer, and 0 points for incorrect answers. All questions are weighted equally.

An undisclosed number of unscored questions will be included on each exam. These are unmarked and indistinguishable from scored questions. Answers to these questions will be used for research purposes only, and will not count towards the score.

Retakes & Cancellations

If you do not pass the exam, you may schedule a retake. You will need to pay a registration fee for each retake. You can reschedule or cancel without penalty on Examity up to your scheduled exam. We will not issue refunds for no-shows.

Accommodations

Please contact Examity with any accommodation requests.

Topic Outline

The **dbt Analytics Engineering Certification Exam** has been designed to assess the following topics and sub-topics.

Topic 1: Developing dbt models

- Identifying and verifying any raw object dependencies
- Understanding core dbt materializations
- Conceptualizing modularity and how to incorporate DRY principles
- Converting business logic into performant SQL queries
- Using commands such as run, test, docs and seed
- Creating a logical flow of models and building clean DAGs
- Defining configurations in dbt_project.yml
- Configuring sources in dbt
- Using dbt Packages
- Utilizing git functionality within the development lifecycle
- Creating Python models
- Providing access to users to models with the “grants” configuration

Topic 2: Understanding dbt models governance

- Adding contracts to models to ensure the shape of models
- Creating different versions of our models and deprecating the old ones
- Configuring model access

Topic 3: Debugging data modeling errors

- Understanding logged error messages
- Troubleshooting using compiled code
- Troubleshooting .yml compilation errors
- Distinguishing between a pure SQL and a dbt issue that presents itself as a SQL issue
- Developing and implementing a fix and testing it prior to merging

Topic 4: Managing data pipelines

- Troubleshooting and managing failure points in the DAG
- Using dbt Cloud
- Troubleshooting errors from integrated tools

Topic 5: Implementing dbt tests

- Using generic, singular, custom, and custom generic tests on a wide variety of models and sources
- Testing assumptions for dbt models and sources
- Implementing various testing steps in the workflow

Topic 6: Creating and Maintaining dbt documentation

- Updating dbt docs
- Implementing source, table, and column descriptions in .yml files
- Using macros to show model and data lineage on the DAG

Topic 7: Implementing and maintaining external dependencies

- Implementing dbt exposures
- Implementing source freshness

Topic 8: Leveraging the dbt state

- Understanding state
- Using dbt retry
- Combining state and result selectors

Sample exam questions

Sample Exam Question 1

One of your models is incremental and contains a lot of data.

You want to test that some columns are not null but do not want to test all the data every time.

Which two are ways to configure dbt to not test the entire table every time?

- [ ] Use the test parameter `limit` to add a limit to the number of rows tested.
- [ ] If your data platform supports it, set a contract on the model with `not_null` constraints on the columns that should not be null.
- [ ] Use the test parameter `error_if` to stop the test as soon as we see that a minimum number of rows failed.
- [ ] Use the test parameter `store_failures` to reuse previous test results.
- [ ] Use the test parameter `where` to add a condition on the data tested.

Explanation: dbt test accepts many configuration parameters but not all of them would avoid testing an incremental model entirely after every run.

The correct answer here is that we could use:
- a `where` parameter on the test to potentially restrict data for a specific set of dates
or
- if supported by the data platform, set a contract on the model with a `not_null` constraint, letting the data warehouse test the new incremental data automatically before it is inserted in the table.

Sample Exam Question 2

Which statement regarding incremental materializations in dbt is true?

- ○ Incremental models are ideal for datasets with millions of rows where rows are only added and never updated.
- ○ Incremental models are always rebuilt upon execution.
- ○ Incremental models must be dropped and rebuilt after schema changes in the upstream model.
- ○ Incremental models never need to leverage the `is_incremental()` macro.
- ○ Incremental models are an appropriate choice for datasets where a large percentage of the dataset is updated each run.

Explanation: If a table consists of a large number of rows and only new data is being added at each run, setting the model as incremental allows dbt to run the transformation only on the new data, saving a lot of time and compute on the data warehouse.
Therefore, the answer is that incremental models are ideal for datasets with millions of rows where data is added but previous rows are not updated.

Sample Exam Question 3

You have been working on your feature branch while your fellow analytics engineers have been merging changes to the head branch.

Which statement regarding keeping your separate branches synced is true?

- ○ You should never merge changes directly to the head branch.
- ○ Use `git fetch` to bring all remote branches to local.
- ○ Use `git pull` to reconcile with the head branch.
- ○ Merge conflicts must be resolved via the dbt Cloud IDE.

Explanation: The correct answer is to use `git pull`. This will actually run both – `git fetch`, downloading the code that has been pushed to the repository, and `git merge` which will combine the code just fetched with the code in the current branch.

Sample Exam Question 4

Consider these models files materialized as tables:

stg_accounts.sql

```
select
...
from salesforce.accounts
```

stg_opportunities.sql

```
select
...
from salesforce.opportunities
```

accounts.sql

```
select
...
from dbt_user.stg_accounts
left join dbt_user.stg_opportunities on account_id
```

When running `dbt run` the models build in the incorrect order.

What should you implement to ensure that the models build in the intended order?

- ○ Update the from clauses of `stg_opportunities` and `stg_accounts` to use the source macro.
- ○ A ref macro in both the from clauses of `stg_accounts.sql` and `stg_opportunities.sql`.
- ○ Change to ephemeral materialization strategy for `stg_opportunities` and `stg_accounts`.
- ○ A ref macro in both the from and join clauses of `accounts.sql`.

Explanation: dbt recognizes dependencies between models when the `ref` macro is used. To ensure that `accounts.sql` model runs after the two other models, every hard coded table in `accounts.sql` must be replaced with the relevant `ref` statement.
Therefore, the answer is that the `ref` macro needs to be set for both the from and join clauses of `accounts.sql`.

Sample Exam Question 5

You want to convert this legacy SQL to dbt models:

```
select

    customer_id,
    customer_name,
    cagg.*

from sample_data.public.customer c
left join (

    select

        customer_id,
        count(order_id) as total_orders,
        min(order_date) as first_order,
        max(order_date) as most_recent_order

    from sample_data.public.orders
    group by customer_id

) cagg
  on c.customer_id = cagg.customer_id
where customer_id in (

  select customer_id

  from sample_data.public.customer c
  left join sample_data.public.nation n
    on c.nation_id = n.nation_id
  left join sample_data.public.region r
    on n.region_id = r.region_id
  where r.region_name = 'EUROPE'

 )
```

How many sources and tables will be defined in `sources.yml`?

- ○ 1 source with 5 tables
- ○ 1 source with 4 tables
- ○ 4 sources with 5 tables
- ○ 4 sources with 4 tables

Explanation: Every source in dbt maps to a database + schema combination. Because the database `sample_data` and schema `public` are used on all references, this is one source. Tables are configured within a source's tables configuration. There are four tables referenced: `customers`, `orders`, `nations`, and `regions`.

Learning Path:

This is not the **only** way to prepare for the exam, but just one recommended path for someone new to dbt to prepare for the exam. Each checkpoint provides a logical set of courses to complete, readings and documentation to digest, and real-world experience to seek out. We recommend this order, but things can be reorganized based on your learning preferences.

Checkpoint 0 - Prerequisites

dbt is a tool that brings together several different technical skills in one place. We recommend starting this path after you've developed foundational git and SQL skills.

For SQL, the exam expects familiarity with **joins, aggregations, common table expressions (CTEs), and window functions.**

For git, the exam expects familiarity with **branching strategies (including development vs main branches), basic git commands, and pull requests.**

Checkpoint 1 - Build a Foundation

Resources:

- Courses:
  - dbt Fundamentals
- Readings:
  - dbt viewpoint
- Documentation:
  - Source properties
  - Node selection syntax
  - dbt_project.yml
  - General resource properties
- Experience
  - Creating a dbt project from scratch to deployment
  - Debugging errors
- Commands
  - `dbt compile`
  - `dbt run`
  - `dbt source freshness`
  - `dbt test`
  - `dbt docs generate`
  - `dbt build`
  - `dbt run-operation`
  - `dbt retry`

Checkpoint 2 - Modularity and Refactoring

Resources:

- Courses:
  - Refactoring SQL for Modularity
- Readings:
  - How we structure our dbt projects
  - Your Essential dbt Project Checklist
- Documentation:
  - Refactoring legacy SQL to dbt
- Experience:
  - Refactoring SQL for performance and clarity

Checkpoint 3 - Doing More with dbt

- Courses:
  - Jinja, Macros, and Packages
  - Advanced Materializations
  - Analyses and Seeds
- Documentation:
  - Exposures
  - Env_var
  - Target
  - Schema
  - Database
  - Grants
  - Model Governance
- Experience
  - Utilizing packages and macros in a dbt project
  - Implementing all core materializations into a dbt project
  - Implementing seeds
  - Being familiar with model access, contracts and versions
- Commands:
  - dbt snapshot
  - dbt seed

Checkpoint 4 - Deployment and Testing

Resources:

- Courses:
  - Advanced Testing
  - Advanced Deployment
- Readings:
  - The exact GitHub pull request template we use at dbt Labs
  - How to review an analytics pull request
  - The “state” selector
- Documentation:
  - Tags
  - Hooks & Operations
  - Custom Schema
  - Threads
- Experience
  - Defining environments in your data platform
  - Defining environments in dbt
  - Promoting code through git including use of multiple branches, pull requests
  - Troubleshooting errors in production runs
  - Defining dbt jobs for optimal performance

Additional Resources:

- dbt Slack
    - #dbt-certification
    - #learn-on-demand
    - #advice-dbt-for-beginners
    - #advice-dbt-for-power-users
    - #dbt-deployment-and-orchestration

Want to use email? Contact certification@dbtlabs.com