
MARC1 MODEL SERIES / CASE STUDY

marc1-Quill

Write and run accurate SQL queries.

Quill by Marc1 writes and runs accurate SQL queries, using analyst-approved examples and QLoRA.

91%

Result accuracy

364 correct first attempts across 400 held-out questions.

62.5%

Fewer errors

Incorrect answers fell from 96 to 36 with identical context.

One GPU

QLoRA training

A compact training workflow with a 24 GB configuration.

[SQL & analytics](#) / [Base: DeepSeek-Coder-6.7B-Instruct](#)

Operations teams ask straightforward questions: how long did a task take, how many tasks were completed, and what changed this week? Answering them accurately requires a model to understand the schema and the business rules behind the numbers. To build marc1-Quill, Marc1 adapted DeepSeek-Coder-6.7B-Instruct for natural-language queries over ClickHouse. Analyst-approved SQL supplied the training targets; execution against reference results supplied the measure of success.

A query can run successfully and still answer the wrong business question. The training target should capture the decisions an analyst would approve.

marc1 / Engineering perspective

01

Business context

Question, schema, and metric definitions

02

marc1-Quill

Approved SQL learned through QLoRA

03

Result evaluation

Execute and compare against approved answers

The model learns from reviewed SQL. The evaluator checks the result of the query.

Understanding the Problem

The difficult SQL errors looked reasonable. Queries executed, returned neatly formatted rows, and still used the wrong denominator, counted the same task twice, or included the wrong reporting day. Analysts repeatedly corrected those decisions.

A compact schema guide helped. Adding business rules and reviewed examples raised result accuracy from 60% to 76%. The team addressed the remaining recurring errors through approved training examples.

Correct syntax is only the beginning. A useful SQL model also needs to know what a completed task means, how a metric is weighted, and which dates belong in the answer.

marc1 / Defining quality

Average handling time made the problem concrete. Averaging daily averages gives a small day the same influence as a busy day. The approved calculation divides total handling seconds by total completed tasks. Both queries can run; only one follows the business definition.

Defining a Correct SQL Result

We curated 1,800 question-context-SQL records. Each paired a natural-language question with versioned schema definitions, business rules, and reviewed SQL. The split happened by query family before paraphrases were added, keeping related variants together. Only training records supplied prompt examples.

1,200	200	400
Training records	Development records	Held-out evaluation records

Related query families stayed within a single split.

Generated SQL ran against fixed development and evaluation snapshots. The evaluator compared the outputs with analyst-approved reference queries, preserving duplicate counts and NULLs, respecting requested ordering, and applying defined numeric tolerances.

Extra fixtures exposed uneven volumes, duplicate events, zero denominators, and date boundaries. A runnable query with the wrong result failed. This execution-based evaluator selected development checkpoints; training itself learned from approved SQL targets. 4

Training on Business Definitions

DeepSeek-Coder-6.7B-Instruct provided a small, code-focused starting point for a bounded SQL task. 1 We concentrated the adaptation on business definitions and recurring errors, using supervised fine-tuning with QLoRA.

The frozen base used four-bit NF4 with double quantization and BF16 computation. Only the LoRA adapters on attention and MLP projections were updated. Hugging Face Transformers, PEFT, and bitsandbytes supplied the training stack. 2 , 3

The training configuration used one BF16-capable GPU with 24 GB of VRAM, rank 16, alpha 32, a learning rate of 1e-4, and two epochs. A microbatch of one, 16-step gradient accumulation, and gradient checkpointing kept memory demand modest.

We used the native instruction template, applied loss only to target SQL tokens, and limited complete records to 2,048 tokens. Overlength examples were shortened without dropping required schema or SQL. The checkpoint with the highest result accuracy on the 200 development questions was then frozen for final evaluation.

Here is an approved target for the question “What was average handling time by project for September 7-13, 2026?” The table has one row per project per UTC day, and both measures cover the same eligible completed tasks.

[Approved SQL / ClickHouse](#)

```
SELECT project_id,  
       toFloat64(sum(handling_s))  
         / nullIf(sum(completed_n), 0) AS aht_s  
FROM project_daily  
WHERE day >= toDate('2026-09-07')  
      AND day <  toDate('2026-09-14')  
GROUP BY project_id  
ORDER BY project_id;
```

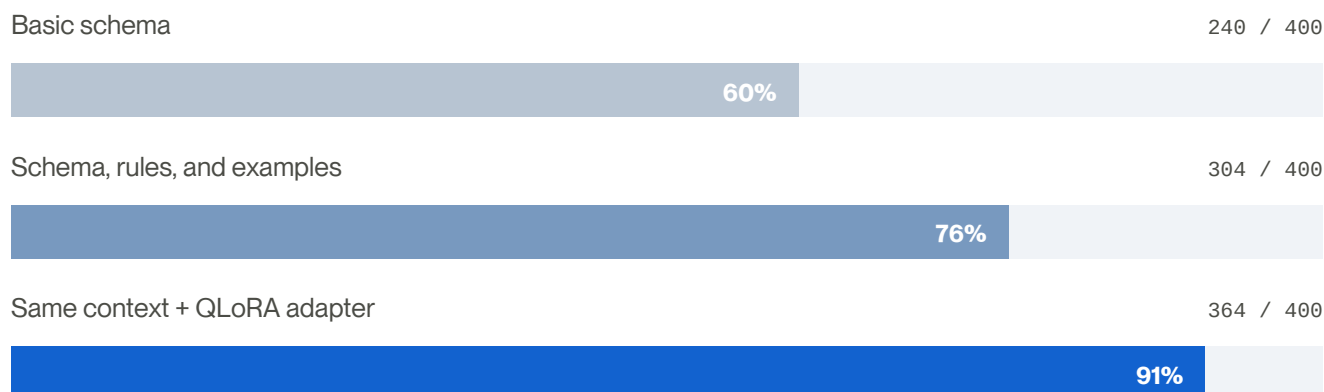
Total handling seconds divided by completed tasks. A zero denominator returns NULL.

In a verification fixture, project A has 600 seconds across two tasks on one day and 1,800 seconds across 18 tasks on another. Averaging the daily averages produces 200 seconds. The approved query returns 120 seconds: 2,400 divided by 20.

The same principle applies to task events. Completion rows T1, T1, and T2 represent two distinct tasks. The approved target uses `uniqExact(task_id)`; counting rows would return three.

Results and the Path to Production

The held-out evaluation covered 400 answerable, in-scope questions across filtering, joins, metric formulas, and time windows. Every configuration used the same four-bit base, fixed database snapshots, greedy decoding, and one attempt with no repairs. The final two configurations used identical prompts and context.

[Result accuracy / 400 held-out questions](#)

With identical context, fine-tuning increased result accuracy from 76% to 91%.

Better context added 16 percentage points. Fine-tuning added a further 15, reducing incorrect answers from 96 to 36: 60 fewer errors and a 62.5% relative reduction.

Fine-tuning improved repeated SQL decisions within the approved schema. The evaluation measured result accuracy on the held-out questions; performance on unfamiliar databases requires separate evaluation.

The application retained analyst review for consequential queries and enforced read-only database access and query limits. Reviewed corrections supplied the next training batch, while a fixed regression suite checked each new adapter before release. Future releases should also use fresh held-out questions to measure generalization.

For marc1, the repeatable workflow is to turn business definitions into approved examples, train on those examples, and evaluate the answers in the setting where the model is used.

Technical sources

1. [DeepSeek Coder model card](#)
2. [QLoRA: Efficient Finetuning of Quantized LLMs](#)
3. [Hugging Face PEFT quantization guide](#)
4. [Semantic Evaluation for Text-to-SQL](#)

Technical references for the base model, QLoRA method, and execution-based evaluation.