Background

Our goal, as summarized in the project PDF and the discussion that followed, is to **teach a 7 B-parameter student model to improve its answers by actively asking clarifying questions** rather than passively copying a teacher's chain-of-thought. The pipeline is therefore split into three logical stages:

1. **Baseline Supervised Fine-Tuning (Track A)** – gives the model a plain yes/no head on StrategyQA.
2. **Chain-of-Thought (CoT) Distillation (Track B)** – injects the teacher's reasoning to boost raw accuracy.
3. **Preference-based Alignment (DPO)** – explicitly rewards answers that emerge *after* the student draft, proving the model benefits from its own questions.

We stay within the 4-bit **QLoRA** memory budget outlined in the PDF, so all three stages run on a single RTX-4060 (8 GB). Diagnostics (ablations and attention attribution) confirm that performance gains vanish when we remove or shuffle the student draft—evidence the model truly learns by asking.

Training / Finetuning Schedule for the Student Model

Phase A – Baseline Supervised Fine-Tune (QLoRA)

Input artifacts
Base model: *7 B Llama-2 weights loaded in 4-bit NF4.*
LoRA config: rank-8, $\alpha = 16$, dropout = 0.05.
Dataset:* `strategyqa_train_baseline.jsonl` – 2 000 rows, each `{question, teacher_final}`.

Procedure
1. Instantiate PEFT `LoraConfig`, wrap the base model with `get_peft_model`.
2. Tokenise **only the question** and target yes/no label (max 128 tokens).
3. Train for **3 epochs** with AdamW (LR 2 e-4, β_1 0.9, β_2 0.98), batch 32, gradient-accumulation = 8.
4. Save checkpoint `models/baseline_phaseA/` containing LoRA adapters + 4-bit weights.

Output / success metric
`baseline_phaseA` *checkpoint*
Dev accuracy baseline $\approx$ 60 % (Report to notebook).

Phase B – CoT Distillation (QLoRA)

Input artifacts
Checkpoint: `baseline_phaseA`.
Dataset: `strategyqa_train_cot.jsonl` – 2 000 rows `{question, teacher_cot, teacher_final}`.

Procedure
1. Load `baseline_phaseA` with frozen 4-bit weights; LoRA adapters remain trainable.
2. Prompt concatenation: `question + SEP + teacher_cot` (<= 128 tokens).
3. Supervised fine-tune for **3 epochs** (same LR/batch).
4. Save checkpoint `models/cot_phaseB/`.

Output / success metric
`cot_phaseB` *checkpoint*
Dev accuracy target +7–10 pp over Phase A (~67–70 %).

Phase C – Alignment via DPO (Cal-DPO + Dr-DPO)

Input artifacts
Checkpoint: `cot_phaseB`.
Preference pairs: `data/pairs.jsonl` with fields `{prompt, chosen, rejected}` where
* `prompt` = `question + " Student draft: " + student_draft`
* `chosen` = teacher_final
* `rejected` = student_draft (or flipped if student already correct).

Procedure
1. Wrap `cot_phaseB` in TRL `DPOTrainer` with **β = 0.1**.
2. **Cal-DPO:** subtract batch-mean log-gap μ each step.
3. **Dr-DPO:** drop top 15 % noisy pairs by disagreement score.
4. Training schedule:
* *Epoch 1* (warm-up) β 0.05, no Cal-DPO.
* *Epochs 2-3* (core) β 0.10, Cal-DPO on.
* *Epoch 4* (robust) β 0.10 + Dr-DPO filter.
5. *Optional* Self-Guided loop: regenerate new draft answers with current model, rebuild pairs, repeat one extra epoch. 6. Save checkpoint `models/dpo_phaseC/`.

Output / success metric
`dpo_phaseC` *checkpoint*
Dev accuracy ≥ +10 pp over Phase A and ≥ +3 pp vs. No-draft ablation
* Validation metrics in notebook: attention attribution ≥ 20 % on draft; GPT-4 question-quality win-rate +10 %.

Validation Experiments (proof the model learns by *asking*)

Test	Procedure	Success Criterion
No-draft ablation	prompt = Q only	≥ 3 pp accuracy drop vs full-draft
Shuffled-draft	pair Q with random draft	further accuracy drop
Attention attribution	Integrated-gradients mass on draft tokens	≥ 20 % after DPO
GPT-4 question-quality	Judge usefulness of clarifiers	+10 % win-rate

---

Compute & Timeline

- **Phase A:** ~2 GPU-h
- **Phase B:** ~2 GPU-h
- **Phase C:** ~3 GPU-h
- **Total:** ≈ 7 h on RTX-4060 (8 GB) – fits 7-day PDF schedule.

Milestones / Merge Requests

MR	Title	Key files
8	Add DPO pair builder	`build_dpo_pairs.py`, sample JSONL
9	Implement Cal-/Dr-DPO trainer	`train_dpo.py`, `configs/dpo.yaml`
10	Add SG-DPO loop & regen script	`iterate_dpo.py`
11	Validation notebook & metrics	`DPO_eval.ipynb`, `attention_tools.py`

This expanded document now includes background context, rationale, and validation criteria derived directly from our conversation and the original project PDF.