GCP exams usually recommend 3+ years of industry experience, including 1+ years of designing and managing solutions using GCP, none of which I had, but you only get to try Beta once, so challenge accepted, and a plan transpired: https://www.meistertask.com/projects/lgkxmr98po/join/
If you want to make edits, please duplicate the project and then remove yourself as a member from the original project, don't archive or edit the original because it affects my copy.
I knew there was time to go through any material for only one pass, so focus and efficiency is crucial, then I discovered creating flashcards using PowerPoint filled with screenshots (~120) of material I have gone through is really helpful to jogging memory. https://drive.google.com/file/d/1fLGQfco8DcTx-g4djL7KQYc2SY1MbT6w/view?usp=sharing

Unfortunately, I did not complete the recommended Machine Learning with TensorFlow on Google Cloud on Coursera and only went through Big Data and Machine Learning Fundamentals, and the first 2 courses (they covered a majority of what's necessary) of Advanced Machine Learning with TensorFlow on Google Cloud Platform Specialization. However, going through the slides of all the other recommended courses was significantly helpful to the exam (especially the 5th course in the advanced specialization). A significant amount of knowledge covered in the exam also came from Google's machine learning crash course.
Most of the preparation tips I have are collected in the MeisterTask planner above already, so I will share my thoughts after taking the exam.

Section 1: ML Problem Framing

Be able to translate the layman language in the question to machine learning terminology, such as what kind of algorithm to use to solve what real-life problems. Read the question carefully, it is not as straightforward as just looking at the label type. Some questions seemed completely business-oriented and required an understanding of business metrics and what is good for the customer.

Section 2: ML Solution Architecture

IAM and permissions may have been implicitly tested through the MCQ options provided, so know what GCP products have additional security features beyond the general IAM. Know what products can be used at each stage (ingest, transform, store, analyze) of a data pipeline. Read very carefully the current state of the company in question and do not choose options that repeat what has already been done by the company or what is too far ahead.
Know the differences between GPU and TPU acceleration and what makes either option impossible or undesirable so the choice is immediately clear once you see the key points in the question. Learn generally about what KMS, CMEK, CSEK do, and how they are used to deal with privacy requirements.

Section 3: Data Preparation and Processing

Be familiar with translating modelling requirements into the right feature engineering steps (hashes, bins, crosses), and hashing for repeatable train-test-split. MLCC (https://developers.google.com/machine-learning/crash-course) is thorough on this. Statistical methods of feature selection should be compared and understood. Quotas and limits are implicitly tested through the options showing substitute products at a particular stage in a pipeline. Knowing common uses of DataFlow vs Cloud Functions would help.
Learn how TFrecords feature in data pipelines and the general ML flow involving them, such as when to convert to them, how to train-test-split with them. Be able to identify data leakage and handle class imbalance (MLCC covers this).

Section 4: ML Model Development

Know the spectrum of the modelling tools on GCP(BQML, SparkMLlib, AutoML, ML API, AI Platform) and their degree of no-code to transfer learning to full custom code. Modelling speed and accuracy are competing requirements. Learn how data/code moves in between GCP ML components and look out for import/export shortcuts and their formats.
Know what kinds of explanations are available in AI Explanations for what types of data.

Section 5: ML Pipeline Automation & Orchestration

Most of the questions were asked at a higher level than I expected, so running through Kubeflow pipelines UI with Qwiklabs, looking at the sample code to see how components connect and understanding how TFX vs Kubeflow differ is sufficient. Note how some things can be done on-prem vs GCP. Learn how to build Kubeflow pipelines fast. There is always a competing concern between no flexibility but fast copy-paste development vs full flexibility but time-consuming development from scratch. Neither is always better, depends on where the company is at in terms of skills and product, and what infrastructure, libraries they currently use or are planning to go towards, so read the question.

Section 6: ML Solution Monitoring, Optimization, and Maintenance

Know the tools to analyze model performance during development. and continuously evaluate model performance in production. Pipeline simplification techniques are introduced in the 2nd course in the Advanced Machine Learning with TensorFlow on Google Cloud Platform Specialization.

General Exam Tips

Some questions are really short that you can answer within 5 seconds. Some time burners exist where the options are longer than the question. Some options can be guessed correctly through careful reading of requirements and common sense. Understand what the question wants and go for the option that does things just right, not more, not less. Some options are a subset of other options. Sometimes the best answer does not fulfil 100% of the question's requirements, but the other options are even more wrong. Sometimes the closest answer suggests you do something undesirable to solve a higher priority problem, so there are elements of sacrifice. There were not many "tick all that is correct" questions. There are general python questions and Tensorflow debugging questions that require real hands-on experience which Qwiklabs will not offer because they can only teach how to succeed, not how to fail.

Read the options first and form a mental decision tree of what are the decision variables to seek from the question. There seems to be very little of the "permute 2 options on 2 decision variables to make up 4 MCQ options", but mainly slightly different options, with up to 4 all correct, but just meeting the requirements at 0–20%, 50%, 70%, 90–100% effectiveness. Some parts of the multi-part options are repeated so there is no need to choose there. Much of the question could be irrelevant once you parse the options so reading the question anymore is wasting time. Filtering out irrelevant options is an effective speed booster. If it's not obvious where the variances in the options are and you have to read the whole question, always start from the big picture of the current state of the company, what stage of the SDLC are they in. If you know the question is talking about deployment, all options regarding development can be eliminated. The options being multi-part could confuse people and make it harder, but it also means there are more opportunities for elimination, so even if you don't understand all the parts of the option, you just need to find one part that makes the whole option wrong.

If time permits, prove not only why your selection is correct, but also why all other options are not on your first pass. If short on time, it is easier to prove options wrong than how the possibly correct one matches all requirements. I had only 24 minutes left to review 58/120 and only managed to review 20.

Handling the Exam UI

Questions load page by page and there are 4 buttons on every page (back, forward, review all, submit). Do not submit until all questions are done and reviewed. The review page will show how many were answered and put an asterisk beside those you marked for review. Have a low threshold for marking review (i had 58/120), because it costs a lot of time to click the back button repeatedly and look for something you did not mark for review but later wanted to. However, if you realize in the middle you don't have 1 min to spare per review, start having a higher threshold for review because having too many asterisks at the end means you could spend time reviewing things you are pretty sure of already rather than on the ones that really need review. The "review all" page only shows you question numbers (with an optional asterisk) and your selection, with no question preview text at all, so unless

you have great memory it's hard to know which number corresponds to which question, so you may have to go through all the asterisks.
Jot down on first pass in the comments box below every question(not sure if this box is only a beta feature) why certain options are wrong so when coming back to review, you don't restart the question from nothing and can immediately think through only the possibly correct competing options. Another use for the comment box is to record concepts you are unsure of. There could be future questions you come across that resolve such uncertainty by providing the answer as a given in the question, such as what tools are used together, which tools call which tools. Google has a history of offering non-existent options, but if you see the same option/concept appearing in more than 1 question, it is likely possible.
Don't click the Back button 2x in succession to prevent accidental submission, because the submit button will be loaded right under your cursor after the 1st click. The back and forward take about 3–5 seconds to load, where the timer stops, so you can get some extra time to think while the page loads. Don't type in CAPS using shift lest you do a Ctrl+C/X or some other combination that gets your exam locked (i got locked twice, luckily I did it onsite so a proctor was there to unlock, not sure how it works if done remotely).

Study Strategy

If you have time

Follow the recommended courses first before going to the tutorials (search ai platform on https://cloud.google.com/docs/tutorials and you cover 95%, the rest are GCS, PubSub, Cloud Functions, Bigquery). The courses cover a majority of what's tested. Another benefit is when you know the concepts already, reading the tutorials will organize the individual tools and concepts into an entire architecture. You can then use the knowledge from tools to ask questions about how a tool performs against another in this architecture, how to stretch its limits, can it be connected to another source/sink, how does 1 tool's quotas/limits affect another tool's limits in the pipeline, where/which tools are the common bottlenecks, where is seldom a bottleneck, where are the serverless parts (2 types: can configure vs no need to configure) and which parts are not serverless.
Opening multiple tools when doing Qwiklabs is useful, such as always keeping the VM console page on, to learn that your 1-click GKE cluster deployment is actually provisioning 3 VMs by default under the hood with certain settings, or that your "Open JupyterLab" click in AI Platform notebooks has provisioned one VM of certain machine type behind the scenes, or that the startup script that was automatically run when you do some Qwiklabs has set up some git clone behind the scenes already. Keeping the GCS console open is important too since so much of GCP AI tools depend on buckets.

If you don't have time

Read the tutorials and documentation (overview, best practice, faq) straightaway. This is the more difficult path because there will be many unknown concepts while going through the tutorials, and they may be too in-depth, that level of knowledge covering < 10% of what's

tested. However, they serve as the fastest starting point for the learner to know the unknowns.

Know the gcloud SDK

https://cloud.google.com/sdk/gcloud/reference

This is the fastest way to know what Google has and how it is named. Expand each section to see the method names and you will have an idea of what services are available without trawling through GCP console UI. This page also alerts you to doc pages you may have missed and helps solve questions that test the correct command to use.

On Day 1, I had totally no idea what 53/81 bullet points in the exam guide meant, or how to achieve those points.
After studying https://developers.google.com/machine-learning/crash-course, I also realized some of the 28/81 which I thought I knew, was not what it's supposed to be.

I don't think having a ton of ML knowledge is necessary for these reasons.

1. The exam has very little implementation/ debugging questions, mostly focusing on GCP tool selection and solution architecting (sometimes open-source tools are given as options but usually GCP tool wins for serverless scalability). I would definitely have not attempted with 20 days of study if any implementation is required.
2. Even if someone did something before (eg. handling imbalanced data), he may not have done it in the google suggested way. Yes, it's not as objective and there are indeed google recommended practices to memorize.
3. A good portion of the exam is on GCP specific tools, commands, workflows. If someone does not study GCP, he won't know what's possible, or how development, test, deploy, monitor workflows are done using GCP tools. Knowing how to do it outside GCP does not mean it's the correct answer. Often on-prem tools or doing it locally is wrong in the exam context.
4. It is not in Google's favour to make exams incredibly hard. People who have enough experience would not need the certificate to prove anything. Making it too hard discourages people from studying for the exam, which means fewer GCP users, fewer exam fees earned, less open-source companies employing these test takers and switching to GCP on a company level.

Some arguments supporting the benefit of previous experience:

1. Dataflow is based on Apache Beam, Cloud Composer on Airflow, AI Platform pipelines on Kubeflow, so if you already used the open-source version, you can go through code in tutorials faster, and know why some tools are overkill and obviously the wrong choice compared to another tool in the multiple-choice. But remember again, implementation is rarely tested. What's more important is knowing what GCP specific source and sinks are available for Dataflow, and how a GCP pipeline allows for certain workflows/shortcuts, which may not have been possible with open source tools.
2. People who read/experience more can better distinguish which business metric to apply for what situation or what ML problem can be framed from given features and vague requirements. However, there is only very basic ML, technical jargon required before common sense can take over.

3. People who read/experience more will know more ways to do something or more ways something can go wrong and its negative impact, and use that knowledge to be able to identify and infer what went wrong when presented a scenario and what steps to take to fix it. (eg. Data leakage, bad train-test-split, training-serving skew, underfitting). However, knowing solutions is not enough, because you must also know what to try first, and here comes again google recommended practices to study.