Perfect — now that you’ve added **Chains** and **Runnables** to your studied topics, I’ll mix them in with your earlier topics for the next batch of advanced questions.

**✅ 10 Conceptual Questions (Advanced Level)**

1. **Explain how LangChain’s SequentialChain differs from building a pipeline with RunnableSequence. Which is more flexible for large-scale workflows, and why?**
2. **What’s the role of RunnableParallel and how does it handle dependencies between outputs?**
3. **Compare SimpleSequentialChain vs. SequentialChain vs. RunnableSequence in terms of control over intermediate steps and output parsing.**
4. **How would you design a chain that integrates both synchronous and asynchronous steps without breaking the flow?**
5. **Describe the key differences between chaining via Runnable objects and chaining via prompt templates directly.**
6. **In what cases would you replace a chain with a custom runnable pipeline, and what benefits/tradeoffs would that bring?**
7. **Explain how error handling is implemented in a runnable pipeline. How would you retry only failed steps?**
8. **How does the RunnableBranch help implement conditional logic in LangChain? Give real-world examples.**
9. **What’s the impact of large intermediate outputs in multi-step chains on performance and cost, and how can you mitigate it?**
10. **Discuss the pros and cons of deeply nested chains vs. flat runnable workflows for long-term maintenance.**

**🚀 30 Programming / Mini Project Questions (High Complexity)**

I’ve combined these with your earlier topics so you get realistic, portfolio-worthy challenges.

**🧠 Multi-Step Reasoning & Chaining**

1. Create a **multi-step reasoning chain** that:
   * Summarizes a legal document
   * Extracts key entities
   * Generates a plain-English explanation for non-experts
   * Uses SequentialChain with a custom output parser.
2. Build a **branching chain** (RunnableBranch) that:
   * Detects if text is financial, medical, or general
   * Routes it to different summarization prompts accordingly.
3. Implement a chain that takes **a messy email** and:
   * Extracts contact details
   * Detects urgency level
   * Sends a structured JSON summary to another chain for storage.
4. Build a pipeline that:
   * Scrapes an article
   * Classifies its domain
   * Summarizes in 3 tones: formal, casual, and marketing style.
5. Create a chain that compares **two LLM-generated answers** and scores them for factual accuracy, tone, and conciseness.

**🔁 Runnable-Based Workflows**

1. Build a RunnableParallel pipeline that:
   * Generates a blog post
   * Creates a title
   * Produces 3 social media captions in parallel.
2. Implement a **multi-model pipeline** where:
   * Step 1 uses GPT for summarization
   * Step 2 uses HuggingFace for sentiment analysis
   * Step 3 formats results into JSON.
3. Create a RunnableMap workflow that takes multiple questions and returns parallel answers for each.
4. Build a **dynamic runnable selector** that chooses between different chains based on input complexity.
5. Implement a chain where **one runnable’s output is validated by another runnable** before returning results.

**⚙️ Integration with Older Topics**

1. Design a chain that:
   * Takes a research abstract
   * Extracts keywords
   * Routes them to a chain that generates related project ideas.
2. Implement a chain with **retry logic** that catches JSON parsing errors and fixes them via a repair prompt.
3. Build a chatbot where **each conversation turn** is processed through a runnable sequence for context management.
4. Create a **prompt quality checker** runnable that scores prompts before sending them to the model.
5. Implement a chain that **uses function calling** for structured data extraction inside a runnable workflow.

**📊 Data Processing & Advanced Parsing**

1. Build a chain that:
   * Processes customer reviews
   * Groups them into categories
   * Produces a JSON report with counts and sentiment score per category.
2. Implement a **structured code explainer**:
   * Takes source code
   * Breaks it into functions/classes
   * Outputs a JSON structure with function names, purpose, and complexity rating.
3. Create a **chat-to-database pipeline** where:
   * User asks questions
   * Output is converted into SQL queries
   * Queries are validated and parsed.
4. Build a **table generator chain**:
   * Takes unstructured text
   * Outputs a structured Markdown or HTML table.
5. Implement a chain that:
   * Extracts key facts from Wikipedia articles
   * Validates them against a known list
   * Outputs only verified facts.

**🧩 Creative & Applied Projects**

1. Create a **role-based assistant pipeline**:
   * Role 1: Brainstorm ideas
   * Role 2: Critique ideas
   * Role 3: Output improved ideas in structured form.
2. Build a **resume optimizer**:
   * Parses a CV
   * Compares it to a job description
   * Outputs a score and improvement suggestions.
3. Implement a **content moderation workflow**:
   * Detects offensive or unsafe content
   * Routes flagged content for human review.
4. Create a **multi-lingual summarizer** that:
   * Detects language
   * Summarizes text
   * Translates summary into 2 other languages.
5. Build a chain that:
   * Takes a product description
   * Generates marketing copy
   * Produces 5 ad headlines and 3 email subject lines.
6. Implement a **debate simulator** runnable:
   * Two LLMs argue opposite sides of a topic
   * A third LLM judges the winner.
7. Create a **fiction plot generator**:
   * Takes a genre and theme
   * Produces an outline, character bios, and sample dialogue.
8. Build a **knowledge extraction tool**:
   * Reads PDF lecture notes
   * Extracts concepts into a mind-map JSON.
9. Implement a **prompt evolution tool**:
   * Takes an initial prompt
   * Automatically improves it based on evaluation metrics.
10. Create a **news fact-checker**:
    * Extracts claims from articles
    * Cross-verifies them via an external API
    * Outputs a fact-check report.

If you keep adding more topics, I’ll keep stacking the difficulty and weaving in old concepts so that by the end, you’ll have **a solid set of 100+ advanced LangChain portfolio projects**.

If you want, I can also **mark which of these are portfolio gold** so you can prioritize them for GitHub. Would you like me to do that for this list?