0 interview questions in each section tailored for an **SDET QA role focused on AI, LLMs, and RAG** systems.

✅ SECTION 1: AI / LLM BASICS

1. What is a Large Language Model (LLM) and how does it work?
2. What is the difference between generative AI and traditional rule-based NLP?
3. How do tokenization and context windows affect LLM responses?
4. What are the differences between zero-shot, one-shot, and few-shot learning in LLMs?
5. Explain the importance of embeddings in LLM-based applications.
6. What are some limitations of LLMs when used in production systems?
7. What is temperature in an LLM API and how does it influence outputs?
8. Describe use cases where LLMs are most effective and where they're not.
9. How is fine-tuning different from prompt engineering?
10. What are common reasons an LLM may return incorrect or biased outputs?

✅ SECTION 2: RAG (Retrieval-Augmented Generation)

1. What is a RAG pipeline and why is it important for grounding LLMs?
2. Explain the flow of data in a typical RAG architecture.
3. What are chunking strategies and why do they matter in RAG?

4. How does a vector database like FAISS or ChromaDB work?
5. What types of embedding models are used in RAG systems?
6. What happens if irrelevant documents are retrieved in RAG?
7. How do you measure retrieval quality in RAG?
8. What are the pros and cons of hybrid search (BM25 + Vector)?
9. How do you test that the retrieved context is used in the final answer?
10. How would you simulate document drift and evaluate its impact on RAG performance?

✅ SECTION 3: LLM Testing & Evaluation

1. What are hallucinations and how do you detect them in LLM outputs?
2. How would you automate the validation of LLM outputs?
3. What evaluation metrics do you use for summarization and question answering?
4. How does RAGAS score answer relevance and faithfulness?
5. What's the difference between semantic similarity and lexical similarity?
6. How can you test the factuality of an LLM-generated answer?
7. What is DeepEval and how is it used in testing?
8. How do you ensure consistent outputs when LLMs are inherently probabilistic?
9. How would you create a test set for an LLM-powered QA bot?
10. What are the limitations of using traditional metrics like BLEU or ROUGE in LLM QA testing?

✅ SECTION 4: Prompt Engineering & Validation

1. What is prompt engineering and why is it important for QA?
2. How do you test for prompt injection vulnerabilities?
3. How do you validate a prompt template dynamically in a CI/CD pipeline?
4. What is few-shot prompting and how does it impact test reliability?
5. How do changes in prompt wording impact LLM outputs?
6. What strategies can prevent prompt misuse or adversarial input?
7. How would you test for sensitive data leakage in LLM responses?
8. What tools can help with A/B testing different prompt templates?
9. How do you test multi-turn conversations with context retention?
10. How do you debug prompt failures when outputs deviate from expectations?

✅ SECTION 5: Automation, Monitoring, CI/CD

1. How would you integrate LLM response testing into a CI/CD pipeline?
2. What kind of alerts would you set up for monitoring a production LLM system?
3. How do you capture and compare outputs over model version upgrades?
4. How do you perform latency benchmarking for LLM APIs?
5. What log data do you collect to debug a failed LLM response?

6. What’s the role of observability in maintaining LLM reliability?
7. How do you ensure cost-efficiency while running automated LLM tests?
8. How would you implement a rollback strategy for an LLM update?
9. What metrics would you expose in Grafana or Prometheus for an LLM app?
10. How do you validate and re-evaluate continuously evolving models and prompts?