1. What does a SavedModel contain? How do you inspect its content?

**A SavedModel contains the model's architecture, weights, and other assets needed for inference.**

**To inspect its content, you can use TensorFlow tools like saved\_model\_cli or load the model in Python using tf.saved\_model.load(), then explore its components, such as input and output signatures.**

2. When should you use TF Serving? What are its main features? What are some tools you can

use to deploy it?

**Use Cases: TF Serving is used for deploying machine learning models for serving predictions, especially in production environments.**

**Main Features:**

**Model Versioning: Easily manage different versions of models.**

**REST and gRPC APIs: Supports both REST and gRPC protocols for querying models.**

**Model Monitoring: Provides monitoring capabilities for deployed models.**

**Scalability: Scales horizontally to handle high request loads.**

**Deployment Tools: You can deploy TF Serving using Docker containers, Kubernetes, or on cloud platforms like Google Cloud AI Platform.**

3.How do you deploy a model across multiple TF Serving instances?

**To deploy a model across multiple instances of TF Serving, you can set up a load balancer that distributes incoming requests to multiple TF Serving servers. This ensures load distribution and high availability.**

4. When should you use the gRPC API rather than the REST API to query a model served by TF

Serving?

**Use the gRPC API when:**

**Low latency and high throughput are critical.**

**You need features like streaming or bidirectional communication.**

**Use the REST API when:**

**Simplicity and ease of integration with existing systems are essential.**

**Interacting with the service from various programming languages.**

5. What are the different ways TFLite reduces a model’s size to make it run on a mobile or

embedded device?

**TFLite reduces model size for mobile/embedded devices by:**

**Quantization: Reducing the precision of model weights and activations.**

**Weight Pruning: Removing low-magnitude weights.**

**Model Optimization Toolkit: Applying various optimization techniques.**

**Post-training quantization: Quantizing weights and activations post-training.**

6. What is quantization-aware training, and why would you need it?

**Quantization-aware training is a technique where a model is trained to be more robust to quantization during inference.**

**It helps maintain model accuracy when using lower-precision data types (e.g., int8) during inference, which is common on mobile and embedded devices.**

7. What are model parallelism and data parallelism? Why is the latter

generally recommended?

**Model Parallelism: Involves splitting a model across multiple devices, with each device handling a portion of the model's layers.**

**Data Parallelism: Involves replicating the entire model on each device and training on different mini-batches of data.**

**Recommendation: Data parallelism is generally recommended because it is easier to implement and scales well across multiple devices.**

8. When training a model across multiple servers, what distribution strategies can you use?

How do you choose which one to use?

**Distribution strategies like MirroredStrategy, CentralStorageStrategy, and ParameterServerStrategy can be used for training across multiple servers.**

**Choosing Strategy: Selection depends on factors like the hardware setup, network bandwidth, and the specific needs of the training workload. MirroredStrategy is commonly used for synchronous distributed training on multiple GPUs.**