Question 1: What type of queues can you create? State their differences.

There are two types of AWS SQS queues:

1. **FIFO Queue** (First-In-First-Out): In this type of queues, the message strings are received in the same order the were sent.

2. **Standard Queue:** Standard queues provide at-least-once delivery, which means that each message is delivered at least once.

More Details differences shown in the pictures (Next Page).

Ref: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html

Question 2: In which situations is a Web-Queue-Worker architecture relevant?

The Web-Queue-Worker architecture is relevant and can be used in many cases, such as:

- Applications with a relatively simple domain.
- Applications with some long-running workflows or batch operations.
- When you want to use managed services, rather than infrastructure as a service (IaaS).

Ref: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/web-queue-worker

Standard queue	FIFO queue
Unlimited Throughput – Standard queues support a nearly unlimited number of API calls per second, per API action (`SendMessage`, `ReceiveMessage`, or `DeleteMessage`). **At-Least-Once Delivery** – A message is delivered at least once, but occasionally more than one copy of a message is delivered. **Best-Effort Ordering** – Occasionally, messages are delivered in an order different from which they were sent.	**High Throughput** – If you use batching, FIFO queues support up to 3,000 messages per second, per API method (`SendMessageBatch`, `ReceiveMessage`, or `DeleteMessageBatch`). The 3000 messages per second represent 300 API calls, each with a batch of 10 messages. To request a quota increase, submit a support request. Without batching, FIFO queues support up to 300 API calls per second, per API method (`SendMessage`, `ReceiveMessage`, or `DeleteMessage`). **Exactly-Once Processing** – A message is delivered once and remains available until a consumer processes and deletes it. Duplicates aren't introduced into the queue. **First-In-First-Out Delivery** – The order in which messages are sent and received is strictly preserved.
	
Send data between applications when the throughput is important, for example: • Decouple live user requests from intensive background work: let users upload media while resizing or encoding it. • Allocate tasks to multiple worker nodes: process a high number of credit card validation requests. • Batch messages for future processing: schedule multiple entries to be added to a database.	Send data between applications when the order of events is important, for example: • Make sure that user-entered commands are run in the right order. • Display the correct product price by sending price modifications in the right order. • Prevent a student from enrolling in a course before registering for an account.