Questions

● How would you deploy this application in production?

To deploy this application in a production setting, I would consider using a service like AWS Lambda for serverless execution. This would make it easier to scale the application as the load increases. I would also consider using AWS Fargate, which is a serverless compute engine for containers and could work well with our Dockerized application. I would use a managed database service like Amazon RDS for Postgres to handle scaling, backups, and updates of the database. This service also allows for easy replication across multiple availability zones, ensuring high availability and data durability.

● What other components would you want to add to make this production ready?

In terms of enhancing this application for production readiness, I would add error handling mechanisms and logging. AWS provides CloudWatch for logging, which can be very useful for tracking and diagnosing application behavior. I would also include health checks and potentially set up alerting for system failures. Finally, I would set up a CI/CD pipeline for the application, ensuring that any changes get tested and deployed in a systematic and reliable manner.

● How can this application scale with a growing dataset?

In regard to scaling with a growing dataset, I would utilize the scalability of AWS services. For instance, I can easily increase the number of Lambda functions as the load increases or use the auto-scaling feature of AWS Fargate if we're using that. For the database, Amazon RDS allows for easy scaling and performance optimization. We can add read replicas to offload read traffic from the main database, and use partitioning and indexing strategies to maintain performance as the data grows.

● How can PII be recovered later on?

Currently, the script uses MD5 Hashing for masking personal identifiable information (PII), which is a one-way function. That is, we cannot retrieve the original data from its MD5 Hash. This method is a common practice to protect PII. However, if there is a business need to retrieve the original PII, a different approach would be needed. One could consider a reversible encryption algorithm. Of course, this would come with additional considerations, such as the secure storage and management of encryption keys.

● What are the assumptions you made?

In creating this solution, I made several assumptions. I assumed that the JSON messages in the SQS queue would be well-formatted and contain all the necessary fields. I also assumed that the app_version field could be converted into an integer for storage in the database. Additionally, I assumed that there would be no rate limits or data caps on the SQS queue or the Postgres database that would prevent the application from reading or writing data quickly. In a real-world scenario, it would be important to validate these assumptions or to build in error handling to account for situations where they don't hold.Currently, the script uses MD5 Hashing for masking personal identifiable information (PII), which is a one-way function. That is, we cannot retrieve the original data from its MD5 Hash. This method is a common practice to protect PII. However, if there is a business need to retrieve the original PII, a different approach would be needed. One could consider a reversible encryption algorithm. Of course, this would come with additional considerations, such as the secure storage and management of encryption keys.