**Quick Writeup**

**File Size and Format Efficiency**

* **Chunk Processing**: we processed large files in parts using Dask.
* **Compression**: we stored the file in a compressed format (gzip) and read it directly using pandas or Dask which support compressed files.
* **Optimized Data Storage**: we can convert the file to a more efficient format like Parquet, which also supports compression and faster reads.

**Output URL List Analysis**

From the Location field, several URL segments are noticed:

* **Base URL changes**: (https://anthembcbsoh.mrf.bcbs.com, https://empirebcbs.mrf.bcbs.com). This might indicate different systems or subdomains handling distinct data sets or regions.
* **Path segments**: Like 2024-05\_301\_, which might indicate date/version and some form of categorization or identification number.

Interpretations:

* Constant segments like the base URL's primary domain suggest centralized hosting or administration.
* Variable segments might represent specific documents or entities, potentially useful for direct data access or reference.

**'Description' Field Utility**

The Description fields, which include names like "Highmark BCBS Western NY PPO" and similar variants, provide:

* **Identifiable information** about the plan or region.
* **Comparison potential** between descriptions for differentiating plan specifics or regions.
* **Completeness and Variability**: Seem reasonably complete for identifying entities but need to be validated for consistency and relevance to specific queries (e.g., distinguishing between Highmark and Anthem).
* **Identify the State**: The State field is a substring in the description field and varies across the document. It can be used to fetch the URL of the business that is likely to be an Anthem NY PPO

**Testing/Deploying/Running Code in Production Environment**

1. **Development and Local Testing**

* **Code Review**: I will have my peers review the code for standard practices, optimization, and potential errors.
* **Unit Testing**: I will write unit tests for each function to ensure they behave as expected. This includes testing the JSON parsing, data transformations, and any custom logic.
* **Integration Testing**: I will test the integration between the different parts of the application, such as the jq command output feeding correctly into the Dask dataframe.

1. **Environment Setup**

* **Dependency Management**: I will also use tools like pipenv or conda to manage libraries and ensure consistency across environments.
* **Configuration Management**: I will use environment variables or configuration files to manage settings (e.g., file paths, API endpoints) that differ between local, staging, and production environments.

1. **Staging Environment Testing**

* **Replicate Production Settings**: The staging environment should closely mimic the production environment. This includes similar hardware configurations, network settings, and external integrations.
* **Performance Testing**: Test how the script performs under expected production load. For large data processing, ensure memory management and processing time are within acceptable limits.
* **Security Testing**: I will verify that the data processing does not expose sensitive data and complies with relevant security policies.

1. **Continuous Integration/Continuous Deployment (CI/CD)**

* **Automated Tests**: I will setup a CI pipeline to run tests automatically when code is pushed to the repository.
* **Deployment Automation**: I will use CD tools to automate the deployment process to staging and production environments upon successful test completions.

1. **Production Deployment**

* **Blue-Green Deployment**: I will mostly onsider using blue-green deployments to minimize downtime and risk by running two identical production environments. Only after the new version is fully tested and verified in the 'green' environment does traffic gradually shift from the 'blue'.
* **Monitoring and Logging**: Also, I should Implement logging and monitoring to track the application's performance and to quickly identify and respond to failures. Tools like Grafana, Prometheus, or ELK stack can be used for monitoring applications and infrastructure.
* **Backup and Rollback Plans**: Ensure there are processes in place for backing up data and rolling back the deployment if necessary.

1. **Documentation and Training**

* **Documentation**: Ensure all aspects of the application and deployment process are well-documented. This includes the setup guide, configuration instructions, and troubleshooting tips.
* **Training**: Provide necessary training to the operations team on managing and troubleshooting the application.

1. **Maintenance and Updates**

* **Patch Management**: Regularly update the software and dependencies to patch vulnerabilities and bugs.
* **Performance Optimization**: Continuously monitor the application's performance and optimize as necessary based on feedback and logs.

1. **Tools and Technologies**

* **Version Control**: Git
* **Testing**: pytest for Python testing
* **CI/CD**: Jenkins, CircleCI, GitHub Actions
* **Containerization**: Docker, potentially Kubernetes for orchestration if scale demands
* **Logging/Monitoring**: ELK Stack, Grafana + Prometheus