Title: Proof of Concept: Data Preprocessing Pipeline for MinIO Buckets at Redback Operations

Project Overview

The goal of this Proof of Concept is to demonstrate the feasibility and functionality of a data pipeline that processes raw CSV files stored in MinIO buckets, performs data cleaning/preprocessing, and uploads the processed data to another MinIO bucket for further analysis. The system is designed to efficiently handle CSV files, address data quality issues, and provide a seamless transition from raw data (Bronze bucket) to processed, clean data (Silver bucket).

Objectives

1. **Verify MinIO Connectivity**: Ensure the MinIO client can successfully connect to the server using provided credentials and list available buckets.
2. **Download Files from Source Buckets**: Retrieve CSV files from designated project buckets (project-2, project-3) and transfer them to the dw-bucket-bronze bucket for raw storage.
3. **Data Preprocessing**: Implement preprocessing steps to clean the raw data by removing empty columns, filling missing values, and standardizing the structure of the dataset.
4. **Upload Preprocessed Files to Silver Bucket**: Upload the cleaned CSV files to the dw-bucket-silver bucket for further analysis.
5. **Automation**: Provide the capability to process multiple files and automate the workflow for future datasets.

Components Involved

- **MinIO**: An object storage solution used to store raw and processed data in structured buckets.
- **MinIO Python Client (minio SDK)**: A Python library used to interact with MinIO, manage files, and move data between buckets.
- **Pandas**: A Python library used for data processing, including handling missing values, removing empty columns, and transforming data.
- **Logging**: Python's logging module is used to track processing progress, errors, and completion.

Steps Demonstrated in PoC

1. MinIO Client Setup

The MinIO client was successfully initialized with environment variables for the host, access key, and secret key. A function was developed to list all available buckets in MinIO, ensuring that the connection is active and functioning.

- **Verification**: Listing the available buckets confirmed connectivity to the MinIO server.

2. File Handling and Transfer

The next step was to download CSV files from the project-2 and project-3 buckets into the dw-bucket-bronze bucket. A naming convention was applied to each file, ensuring consistency in file naming based on the project name, dataset description, and a timestamp.

- **Verification**: Files were successfully transferred from the source buckets to the Bronze bucket with the correct naming convention.

3. Data Preprocessing

CSV files from the Bronze bucket were pre-processed by:

- **Removing Empty Columns**: Columns that had no data were automatically dropped.
- **Handling Missing Values**: Numeric columns with missing values were filled with the column's median value.
- **Header Validation**: For files with potential missing headers, the first row was used as a header, and preprocessing was adjusted accordingly.
- **Verification**: Preprocessing successfully cleaned the data, ensuring it was ready for further use.

4. File Upload to Silver Bucket

The processed files were then uploaded to the dw-bucket-silver bucket for storage. This separates the raw, uncleaned data (Bronze) from the cleaned and ready-to-use data (Silver).

- **Verification**: Processed files were successfully uploaded to the Silver bucket with the correct file structure.

5. Error Handling and Logging

The PoC includes robust error handling for common issues, such as failed file downloads or invalid CSV structures. Logging is used throughout to track progress, flag errors, and confirm successful uploads.

- **Verification**: The system successfully logged key actions, and any issues during file transfer or processing were handled with appropriate warnings or errors.

Challenges Identified

- Handling files without headers required additional validation and preprocessing logic.
- Some datasets required more complex handling of missing values, which could lead to additional customization in the pipeline.

Next Steps

- **Testing at Scale**: Validate the system's functionality with larger datasets and more files to ensure that the pipeline can handle high-volume operations efficiently.
- **Automation**: Automate the process to continuously monitor the source buckets and automatically trigger preprocessing and uploads when new files are added.
- **Security Enhancements**: Implement access controls to ensure that only authorized users can upload or modify files in the buckets.

Conclusion

The Proof of Concept successfully demonstrates that the data preprocessing pipeline can be implemented in MinIO using Python. CSV files can be downloaded, cleaned, and moved between buckets efficiently, with clear logging and error handling in place. The pipeline is ready for scaling, automation, and integration into broader data workflows at Redback Operations.