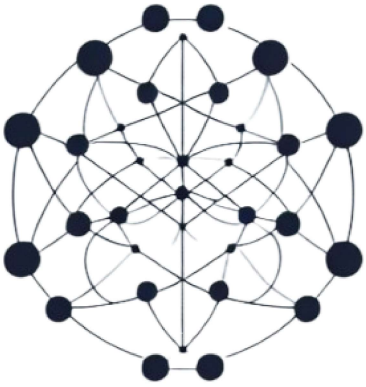
Group Name



data colabs

System Documentation

For

Final Project Work

Document Version:1.0

Date: 14/05/2024

data colabs

Contents

1 Document Management

1. Project Overview: Provide a brief introduction to the project, including its objectives, scope, and relevance.
2. Project Structure: Outline the structure of the project, including the main scripts/modules used, their functionalities, and how they interact with each other.
3. Data Acquisition: Describe the process of acquiring data from Binance, including the tools, APIs, or web scraping techniques used.
4. Data Modeling: Explain how the acquired data is cleaned, transformed, and modelled to prepare it for analysis.
5. Data Persistence: Detail how the cleaned and modelled data is stored and managed, including any compression techniques or cloud storage services used.
6. Data Warehousing: Describe the process of loading data into a data warehouse (e.g., Redshift) for efficient storage and analytics.
7. Data Consumption: Explain how users can access and consume the data for analysis, visualisation, and other purposes.
8. System Design: Provide an overview of the system design, including its scalability, reliability, and performance considerations.
9. Technical Considerations: Discuss specific technical considerations, such as libraries, tools, configurations, error handling, logging, and security measures.
10. Trade-offs and Rationale: Document the trade-offs made during the project and the rationale behind key design decisions, including why Binance was chosen as the data source.
11. Conclusion: Summarise the key points of the documentation and highlight any future considerations or improvements.

Writing Style

- Use clear and concise language to explain technical concepts.
- Provide examples, code snippets, and diagrams where necessary to enhance understanding.
- Use headings, subheadings, and bullet points to organise information effectively.
- Include references and citations for any external resources or libraries used.
- Ensure consistency in terminology and formatting throughout the document.

Review and Feedback

- Have the document reviewed by team members and stakeholders to ensure accuracy and completeness.
- Incorporate feedback and revisions as necessary to improve clarity and understanding.
- Update the document regularly to reflect any changes or updates to the project.

1.1 *Contributors*

Please provide details of all contributors to this document

Role	Name
Project Manager	Abigail
Data Engineer , Project Coordinator	Slyvester
Data Engineer & Cloud Architect	Ishmael
Data Engineer, Software Developer , Technical Writer	Justice
Quality Assurance , Data Engineer	Peter

1.2 *Version Control*

Please document all changes made to this document since initial distribution.

Date	Version	Author	Section	Amendment
14/05/2024	1.0	Data Group	1.0	

2 OVERVIEW

The Binance Data Pipeline project is designed to automate the end-to-end process of collecting, cleaning, modelling, persisting, warehousing, and consuming data from Binance, a leading cryptocurrency exchange platform. The primary goal is to create a robust and scalable data infrastructure that supports data-driven decision-making, analysis, and insights generation for various stakeholders.

Key Components:

1. **Data Acquisition:** The project starts with data acquisition, where data is collected from Binance using APIs, web scraping techniques, or other data extraction methods. This step ensures a continuous flow of real-time or near-real-time data into the pipeline.
2. **Data Cleaning:** Once the data is acquired, it undergoes thorough cleaning and preprocessing. This involves handling missing values, removing duplicates, standardising formats, and addressing any data quality issues to ensure the integrity and accuracy of the data.
3. **Data Modeling:** Cleaned data is then transformed and modelled to extract meaningful insights. This step includes feature engineering, data aggregation, statistical analysis, and the application of machine learning algorithms for predictive modelling and trend analysis.
4. **Data Persistence:** The cleaned and modelled data is persisted in a secure and scalable storage environment. This may involve compressing the data, encrypting sensitive information, and uploading it to cloud storage services such as Amazon S3 for efficient data management.
5. **Data Warehousing**: The persisted data is loaded into a data warehouse, such as Redshift, for structured storage, querying, and analysis. This step involves defining schemas, optimising data structures, and managing data partitions for efficient data retrieval and processing.
6. **Data Consumption:** The final step involves data consumption, where stakeholders, such as data analysts, researchers, and decision-makers, access the stored data for analysis, visualisation, reporting, and decision-making purposes. This may involve building dashboards, generating reports, and conducting ad-hoc queries.

Technical Infrastructure:

- **Cloud Services:** The project leverages cloud services, such as AWS or Azure, for scalable computing power, storage, and data management capabilities.
- **Data Pipeline Automation:** Automation tools and frameworks, such as Apache Airflow or AWS Glue, are used to automate data pipeline workflows, schedule tasks, and monitor data processing activities.
- **Security Measures**: Robust security measures, including encryption, access controls, and data masking techniques, are implemented to protect sensitive data and ensure regulatory compliance.
- **Scalability and Performance:** The infrastructure is designed to be scalable and performant, capable of handling large volumes of data, high concurrency, and complex analytical queries.

Role Assignments:

1. **Project Manager**: Oversees project planning, resource allocation, and coordination among team members and stakeholders.
2. **Data Engineer:** Designs and implements the data pipeline infrastructure, including data acquisition, cleaning, modelling, and persistence components.
3. **Data Scientist:** Applies statistical analysis, machine learning techniques, and data modelling algorithms to derive actionable insights from the data.
4. **Database Administrator (DBA):** Manages database systems, ensures data integrity, performance optimization, and implements data security measures.
5. **Software Developer:** Develops custom scripts, applications, and tools to automate data processing tasks, integrate systems, and enhance data pipeline functionalities.
6. **Cloud Architect**: Designs and manages cloud infrastructure, ensuring scalability, reliability, and cost-effectiveness of the data pipeline.
7. **Security Specialist**: Implements data security measures, access controls, encryption, and compliance standards to protect data assets.
8. **Quality Assurance (QA) Engineer**: Conducts testing, validation, and performance tuning to ensure the quality and reliability of the data pipeline.
9. **Business Analyst**: Gathers business requirements, defines data needs, and translates business objectives into technical specifications for the data pipeline.
10. **Technical Writer:** Creates and maintains technical documentation, user guides, and training materials for the data pipeline project.

2.1 Service Description

The Binance Data Pipeline service automates the collection, cleaning, modelling, persistence, warehousing, and consumption of data from Binance, a cryptocurrency exchange platform. It comprises several major components and business processes that work together to deliver actionable insights and support data-driven decision-making.

2.2 Data Model

Data Model for Binance Data Pipeline

1. **Data Acquisition Model:**
    - Data Source: Binance API or Web Scraping Tool
    - Data Attributes: Timestamp, Trade Volume, Price, Currency Pair, Trade Type (Buy/Sell), Trade ID, Trade Quantity, Market Depth, Order Book Updates, etc.
    - Data Acquisition Process: Fetch data in JSON format from Binance API endpoints or scrape data from Binance website using web scraping tools.

- Data Storage: Temporary storage for raw data before cleaning and processing.

2. **Data Cleaning Model:**
   - Data Cleansing Techniques: Handling missing values, removing duplicates, standardising data formats, correcting data anomalies, and ensuring data consistency.
   - Data Quality Checks: Conducting data quality checks to identify and rectify errors in the data.
   - Data Cleaning Process: Transforming raw data into a clean and standardised format suitable for analysis and modelling.
   - Cleaned Data Attributes: Timestamp, Cleaned Price, Cleaned Trade Volume, Cleaned Trade Type, Cleaned Currency Pair, Cleaned Trade Quantity, Cleaned Market Depth, Cleaned Order Book Updates, etc.
   - Cleaned Data Storage: Storing cleaned data in a structured format for further processing.
3. **Data Modeling and Analysis Model:**
   - Data Transformation: Aggregating data, creating derived features, applying statistical techniques, and building machine learning models.
   - Data Modeling Techniques: Regression, Classification, Time Series Analysis, Clustering, etc.
   - Modelled Data Attributes: Predicted Price, Predicted Trade Volume, Trend Analysis Results, Statistical Metrics, etc.
   - Model Training and Evaluation: Training machine learning models on historical data, evaluating model performance, and fine-tuning algorithms.
   - Insights Generation: Generating actionable insights, trends, patterns, and anomalies from the modelled data.
4. **Data Persistence Model:**
   - Persistent Storage: Amazon S3 Bucket or Similar Cloud Storage
   - Data Compression: Compressing data (e.g., using ZIP format) to reduce storage costs and optimise data transfer.
   - Data Encryption: Encrypting sensitive data to ensure data security and privacy.
   - Archival Strategy: Implementing data retention policies and archival strategies for long-term storage.
5. **Data Warehousing Model:**
   - Data Warehouse: Amazon Redshift or Similar Cloud-Based Data Warehouse
   - Schema Design: Designing schemas for structured storage and efficient querying.
   - Data Loading: Loading cleaned and modelled data into the data warehouse for analysis and reporting.
   - Data Partitioning: Partitioning data for faster query performance and resource optimization.
   - Indexing: Indexing tables for faster data retrieval and query optimization.
6. **Data Consumption Model:**
   - Data Access: Providing access to stored data for stakeholders via APIs, SQL queries, or visualisation tools.

- Data Visualization: Creating dashboards, charts, and reports for data visualisation and analysis.
- Ad-Hoc Querying: Enabling ad-hoc querying for on-the-fly data analysis and exploration.
- Decision Support: Supporting data-driven decision-making by providing timely and accurate data insights.

2.3 Reporting

1. **Reporting Platform:**
   - Reporting is provided via a dedicated Business Intelligence (BI) solution integrated with the Binance Data Pipeline system.
   - The BI solution used is the University-based BI Suite, which includes Web Intelligence (Webi) and Explorer for reporting and data exploration.
2. **BI Suite Details:**
   - Universe: The reporting is based on a custom-built universe designed specifically for the Binance Data Pipeline project. The universe includes data entities and relationships relevant to cryptocurrency trading, such as trade volume, price trends, market depth, and order book updates.
   - Folder and Access Restrictions: Access to reports and folders within the BI suite is restricted based on user roles and permissions. Data sensitivity and confidentiality are considered when defining access restrictions.
3. **Data Source for Reporting:**
   - Reporting is performed against the data warehouse (e.g., Amazon Redshift) where cleaned, modelled, and aggregated data from the Binance Data Pipeline is stored.
   - Data extraction from the transactional database is not performed directly for reporting to avoid impacting transactional processes.
4. **Extract Process Details:**
   - Frequency: Data is extracted on a regular basis, typically daily or hourly, depending on the data freshness requirements of reports.
   - Automation: Data extraction is automated using scheduled jobs within the BI suite or ETL (Extract, Transform, Load) processes.
   - Aggregation and Data Transformation: Extracted data undergoes aggregation, summarization, and transformation to generate meaningful insights for reporting purposes.
5. **User Population for Reporting:**
   - The user population for reporting includes:
     - Data analysts and researchers analysing market trends, trade patterns, and cryptocurrency performance.
     - Business stakeholders and decision-makers using reports for strategic planning, risk assessment, and investment decisions.
     - Compliance officers and regulatory teams monitoring transactional activities and ensuring regulatory compliance.
6. **Automated Jobs for Reporting:**

- Scheduled Jobs: Automated jobs are scheduled to extract, transform, and load data into the reporting environment at specified intervals (e.g., daily, hourly).
- Error Reporting: Automated error reporting mechanisms are in place to notify administrators or designated users of any data extraction, transformation, or loading errors. Error logs are maintained for troubleshooting and resolution.

2.4 Technology

1. **Programming Languages:**
   - Python: Used for scripting data acquisition, cleaning, modelling, and automation tasks.
   - SQL: Used for querying databases, data manipulation, and data warehouse operations.
2. **Frameworks and Libraries:**
   - Pandas: Python library for data manipulation, cleaning, and transformation.
   - NumPy: Python library for numerical computing and array operations.
   - SciPy: Python library for scientific computing and statistical analysis.
   - Scikit-learn: Python library for machine learning algorithms and predictive modelling.
   - SQLAlchemy: Python library for SQL database interaction and ORM (Object-Relational Mapping) capabilities.
   - Boto3: Python library for interacting with Amazon Web Services (AWS) services such as S3 and Redshift.
   - Apache Airflow: Workflow automation tool for scheduling and managing data pipeline tasks.
   - Matplotlib and Seaborn: Python libraries for data visualisation and plotting.
   -
3. **Cloud Services:**
   - Amazon Web Services (AWS):
     - Amazon S3: Cloud storage for storing raw, cleaned, and modelled data.
     - Amazon Redshift: Cloud-based data warehouse for structured data storage and querying.
4. **Database Technologies:**
   - PostgreSQL: Used as the relational database for storing transactional data and intermediate data processing.
   - Redshift: Cloud data warehouse for scalable and efficient data storage and analysis.
5. **Reporting and Business Intelligence Tools:**
   - University-based BI Suite: Web Intelligence (Webi) and Explorer for reporting, data exploration, and visualisation.
   - Excel and CSV exports for ad-hoc reporting and data analysis.
6. **Development Methodologies:**

- Agile Development: Agile methodologies such as Scrum or Kanban for iterative development, collaboration, and flexibility in project management.
- DevOps Practices: Continuous Integration (CI) and Continuous Deployment (CD) practices for automated testing, deployment, and monitoring of the data pipeline.
- Test-Driven Development (TDD) and Quality Assurance (QA) processes for ensuring code quality, reliability, and performance.

7. **Version Control and Collaboration Tools:**
    - Git and GitHub: Version control system and collaborative platform for code management, versioning, and team collaboration.
    - Jira: Project management tool for Agile development, issue tracking, and task management.

2.5 Development Tools

1. **IDEs and Development Environments:**
    - PyCharm: Integrated Development Environment (IDE) for Python development, providing code editing, debugging, and version control features.
    - Jupyter Notebook: Interactive notebook environment for data exploration, analysis, and prototyping Python scripts.
    - Visual Studio Code: Lightweight code editor with Python support for script editing and debugging.
2. **Source Control and Versioning:**
    - Git and GitHub: Version control system and collaborative platform for code management, versioning, and team collaboration.
        - Branching Strategy: Feature branches for development, master/main branch for stable releases, and pull requests for code review and merging.
        - Commit Messages: Descriptive commit messages following conventions (e.g., semantic versioning) for clear version tracking.
        - Continuous Integration (CI): Integration with CI tools (e.g., Jenkins, Travis CI) for automated testing and build validation.
3. **Automated Deployment:**
    - Continuous Deployment (CD): Automated deployment pipelines using CI/CD tools (e.g., Jenkins, GitLab CI/CD) for seamless deployment of code changes to development, staging, and production environments.
    - Docker and Kubernetes: Containerization and orchestration tools for packaging applications, managing dependencies, and deploying microservices.
4. **Local Environment Setup for Debugging:**
    - Virtual Environments: Using Python virtual environments (e.g., virtualenv, conda) to create isolated development environments for installing dependencies and managing packages.
    - Docker Compose: Defining local development environments with Docker Compose for replicating production-like setups locally.
    - Debugging Tools: Utilising built-in debugging features of IDEs (e.g., PyCharm, Visual Studio Code) for step-by-step debugging, breakpoints, variable inspection, and logging.
5. **Testing and Quality Assurance:**
    - Unit Testing: Writing unit tests using Python testing frameworks (e.g., pytest) for code coverage, regression testing, and quality assurance.
    - Automated Testing: Integration with automated testing tools (e.g., Selenium for web scraping testing) to ensure data pipeline functionality and reliability.
    - Code Reviews: Conducting code reviews through pull requests on GitHub for code quality, best practices, and knowledge sharing.
6. **Documentation and Collaboration:**
    - Markdown: Writing project documentation using Markdown format for README files, documentation pages, and project guidelines.
    - **Wiki Pages:** Utilising GitHub Wiki pages or Confluence for detailed project documentation, architecture diagrams, and development guides.

- **Collaboration Tools**: Using communication and collaboration tools (e.g., Slack, Microsoft Teams) for team communication, updates, and discussions.

2.6 Interfaces and services

1. **Incoming Interfaces:**
   - Binance API: Incoming interface for real-time data acquisition from Binance cryptocurrency exchange platform.
   - Web Scraping Tools: Interface for scraping data from external sources (e.g., news websites, social media) for additional data enrichment.
2. **Outgoing Interfaces:**
   - Reporting and Visualization Tools: Outgoing interfaces for exporting data to reporting and visualisation tools (e.g., University-based BI Suite, Excel, CSV) for analysis and presentation.
   - Automated Alerts: Outgoing interfaces for sending automated alerts, notifications, and reports to stakeholders based on predefined triggers or thresholds.
3. **Interface Setup:**
   - Data Acquisition (Binance API): Pull interface setup where data is pulled from Binance API on demand or based on predefined schedules (e.g., hourly, daily).
   - Web Scraping Tools: Pull interface setup where web scraping tools are triggered to pull data from external sources periodically (e.g., nightly) or on demand.
   - Reporting and Visualization Tools: Push interface setup where cleaned and modelled data is pushed to reporting and visualisation tools for real-time or scheduled reporting.
   - Automated Alerts: Push interface setup where automated alerts and notifications are pushed to stakeholders based on predefined conditions or events.
4. **Web Services and APIs:**
   - Binance API: Web service interface provided by Binance for accessing real-time cryptocurrency market data, trade history, and order book updates.
   - Internal APIs: Custom-built APIs within the application for data transformation, aggregation, modelling, and data access by reporting tools and external systems.
   - **Reporting APIs:** APIs exposed for reporting and visualisation tools to access cleaned, modelled, and aggregated data for generating reports, charts, and dashboards.

2.7 Access, Authentication and Authorisation

1. **Access to the Application:**
    - Web-Based Access: The Binance Data Pipeline application is accessed through a web-based interface hosted on a designated server. The URL for accessing the application is [URL Here].
2. **Bespoke Access Requirements**:
    - Client-Server Architecture: For bespoke access, clients require the following:
        - Secure Network Connection: Clients must have a secure network connection (VPN or HTTPS) to access the application server.
        - Access Credentials: Clients are provided with unique access credentials (username/password or API keys) for authentication and authorization.
        - System Compatibility: Clients' systems must meet compatibility requirements (e.g., browser version, operating system) to ensure smooth application functionality.
        - Data Access Permissions: Clients' access permissions are defined based on their role and responsibilities within the application (e.g., data analysts, administrators, researchers).
3. **Authentication Process:**
    - User Authentication: The authentication process verifies the identity of users accessing the application.
        - Username/Password: Users provide their username and password credentials for authentication.
        - Two-Factor Authentication (2FA): Optional two-factor authentication may be implemented for enhanced security, requiring users to provide a second form of verification (e.g., OTP sent to a mobile device).
    - Token-Based Authentication: API access may require token-based authentication using API keys for secure access to data endpoints.
4. **Authorization Process:**
    - Role-Based Access Control (RBAC): The authorization process determines the level of access and permissions granted to authenticated users based on their roles and responsibilities.
        - User Roles: Roles such as Admin, Data Analyst, Researcher, and Viewer are defined with specific access permissions.
        - Access Control Lists (ACLs): Access to data, reports, and functionalities is controlled through ACLs configured for each role.
    - Fine-Grained Authorization: Granular permissions may be set for specific actions, data sources, or modules within the application based on user roles and business requirements.
5. **Security Measures:**
    - Data Encryption: Secure Socket Layer (SSL) encryption is used to encrypt data transmitted between clients and the application server to prevent unauthorised access.
    - Access Logging: Logging access activities, login attempts, and authentication events to monitor and audit user access for security compliance.

- Session Management: Implementing session timeouts, session tokens, and secure cookie handling to manage user sessions securely.
- IP Whitelisting: Optionally, IP whitelisting may be implemented to restrict access to specific IP addresses or ranges for added security.

2.8 Delivery

The Binance Data Pipeline application is delivered as a standalone application accessible via a dedicated portal hosted on a designated server. It is not embedded or part of a wider application such as MyEd or any other integrated platform. Users access the application through a web-based interface using the provided URL [URL Here]. The portal provides a centralised platform for users to interact with the data pipeline, access reports, visualise data, and perform data analysis tasks related to cryptocurrency market data from Binance.

3 Support details

3.1 Third Party

The Binance Data Pipeline project involves third-party tools, services, and libraries to enhance functionality and support specific functionalities. Below are details of third-party involvement and relevant information:

1. **Third-Party Tools and Services:**
    - Python Libraries: Various Python libraries such as Pandas, NumPy, SciPy, and Scikit-learn are utilised for data manipulation, analysis, and machine learning tasks.
    - Boto3: Python library for interacting with Amazon Web Services (AWS) services such as S3 and Redshift for data storage and warehousing.
    - Apache Airflow: Workflow automation tool used for scheduling and managing data pipeline tasks.
2. **Third-Party Contacts:**
    - Python Libraries: These libraries are open-source and maintained by their respective communities. Support and technical information can be found on their official documentation websites and community forums.
    - Boto3: AWS provides comprehensive documentation, support forums, and technical resources for Boto3 library usage. Updates and new features are announced through AWS blogs and documentation.
    - Apache Airflow: Apache Airflow is an open-source project with active development and community support. Technical information, updates, and documentation can be found on the Apache Airflow official website and community forums.
3. **Software or Source Code Download Information:**
    - Python Libraries: Source code for Python libraries can be downloaded using package managers such as pip or conda. Documentation and installation instructions are available on their official websites.
    - Boto3: Boto3 is installed as a Python package and can be downloaded using pip or included in project dependencies.
    - Apache Airflow: Apache Airflow source code and installation instructions can be found on the Apache Airflow official website and GitHub repository.
4. **Frequency of Updates and Upgrade Process:**
    - Python Libraries: Updates for Python libraries are released periodically by the respective development communities. Users can check for updates using package managers and follow upgrade instructions provided in the documentation.
    - Boto3: AWS updates Boto3 to align with AWS service updates and new features. Users are notified of updates through AWS documentation and blogs. Upgrading Boto3 follows standard Python package upgrade procedures.
    - **Apache Airflow:** Apache Airflow releases updates and new versions regularly. Users can check for updates on the official website and GitHub repository. Upgrading Apache Airflow involves following

upgrade guides and migration instructions provided in the documentation.

3.2 Documentation

For comprehensive documentation related to the Binance Data Pipeline project, including user guides, technical specifications, and third-party supplier documentation, please refer to the following links:

1. **User Documentation:**
    - Link to User Documentation: This documentation provides guidelines, instructions, and best practices for users interacting with the Binance Data Pipeline application. It covers user roles, interface usage, data visualisation, and reporting functionalities.
2. **Technical Documentation:**
    - Link to Technical Documentation: The technical documentation offers detailed insights into the architecture, design rationale, data flow, API integrations, data modelling techniques, and system configuration of the Binance Data Pipeline project. It is intended for developers, system administrators, and technical stakeholders.
3. **Supplier Documentation:**
    - Link to Supplier Documentation (Boto3, Python Libraries, Apache Airflow, etc.): Supplier documentation includes technical guides, API references, release notes, upgrade instructions, and support resources provided by third-party suppliers such as Boto3, Python Libraries, Apache Airflow, and other integrated tools and services used in the project.

Note(To be linked to github to follow the README.md to be able to run app)

3.3 Standard tasks

1. **Manual Intervention Tasks:**
    - Data Quality Checks: Periodic manual data quality checks are required to ensure data accuracy, completeness, and integrity. This includes

reviewing data anomalies, outliers, and inconsistencies that may not be automatically detected by the system.
    - Exception Handling: Handling exceptions and edge cases that require human intervention, such as resolving data validation errors, addressing API rate limit exceeded issues, or handling data source changes.
    - Data Validation and Verification: Manual validation and verification of data transformations, aggregations, and model outputs to validate results against expected outcomes and business rules.
2. **Regular Business Cycle Tasks:**
    - Annual Data Roll Overs: Performing annual data roll overs or archiving historical data to maintain database performance and manage data storage costs. This involves migrating older data to archival storage or creating snapshots for long-term retention.
    - Data Clean-Up and Maintenance: Regular data clean-up tasks to remove redundant, obsolete, or outdated data, ensuring database efficiency, query performance, and data integrity.
    - Manual Data Extracts: Extracting specific data subsets or custom reports manually for ad-hoc analysis, regulatory reporting, or stakeholder requests that cannot be automated due to unique requirements.

3.4 Troubleshooting Guide

This troubleshooting guide provides details on common issues, error handling, debugging instructions, known problems, and steps to address potential errors in the Binance Data Pipeline application.

1. **Known Accessibility Issues:**
    - Browser Compatibility: The application is optimised for modern web browsers (e.g., Chrome, Firefox, Edge). Known issues with older browser versions may affect functionality.
2. **Finding Error Messages:**
    - Error Logs: Check application logs and error messages for detailed information on encountered issues. Logs can be found in [specific directory or file path].
3. **Special Debugging Instructions:**
    - Debug Mode: Enable debug mode in development environments to track code execution, variable values, and identify potential bugs.
    - Logging: Implement detailed logging throughout the application to capture events, errors, and data processing steps for troubleshooting.
4. **Predicted Errors and Likely Causes:**
    - Data Integrity Issues: Predicted errors related to data integrity (e.g., missing values, data format inconsistencies) may occur due to data source changes or input errors.

- API Limitations: Errors related to API rate limits, network connectivity issues, or server downtime may impact data acquisition and processing.

5. **Identifying Offending Data Items:**
    - Data Validation Scripts: Use data validation scripts or custom tools to identify offending data items causing data-related errors.
    - Fixing Errors: Address data issues by correcting input data, updating data transformation logic, or implementing data validation checks in the pipeline.
6. **List of Known Problems:**
    - Unresolved Issues: Some known problems may not have been fixed in this version. Refer to the issue tracker or documentation for a list of known issues and workarounds.
7. **Handling Failed Scheduled Tasks:**
    - Manual Task Execution: For scheduled tasks that have failed, they can be run manually by triggering the task from the administration panel or command line.
    - Implications of Daytime Execution: Considerations for running tasks during the day include system load, resource utilisation, and potential impact on real-time data processing.
    - Clean Up Work: Before retrying failed tasks, ensure any incomplete or erroneous data is addressed, and perform necessary clean-up operations to avoid data duplication or inconsistency.

Troubleshooting Steps:

1. **Identify Error Type:**
    - Determine the nature of the error (e.g., data-related, connectivity, API error).
2. **Review Error Logs:**
    - Check application logs for error messages, timestamps, and stack traces to pinpoint the issue.
3. **Debugging Mode:**
    - If available, enable debug mode to trace code execution and identify potential bug locations.
4. **Data Validation:**
    - Use data validation scripts or tools to identify and fix data integrity issues.
5. **Retry Failed Tasks:**
    - For scheduled task failures, retry manually after addressing any underlying issues or dependencies.
6. **Collaborate with Development Team:**
    - Communicate with the development team to report and resolve persistent issues, seek guidance, and implement long-term solutions.

4 Document Sign Off

Please add other sign off roles where required:

Role	Name	Date of Sign off
Project Manager	Abigai	
Data Engineer , Project Coordinator	Slyvester	
Data Engineer & Cloud Architect	Ishmael	
Data Engineer, Software Developer , Technical Writer	Justice	
Quality Assurance , Data Engineer	Peter	