**Project Scenario: Data Integration for Sales Data**

You have been hired as a data integration developer for a retail company that operates in multiple locations and collects sales data from various sources. The company wants to integrate sales data from different sources into a central data repository for analysis and reporting purposes.

Your task is to use Python, a popular programming language for data integration, to design and implement data integration workflows to extract data from different sources, transform it into a common format, and load it into the central data repository.

**Requirements:**

1. **Data Sources**: The company has the following data sources that need to be integrated into the central data repository:

• **Point of Sale (POS) System**: Data about sales transactions including transaction ID, customer ID, product ID, quantity, price, and transaction date.

• **Online Sales Platform**: Data about online sales including order ID, customer ID, product ID, quantity, price, and order date.

• **Inventory System:** Data about product inventory including product ID, product name, category, price, and stock quantity.

2. **Data Integration Workflow**: Design and implement a data integration workflow using Python to extract data from the above data sources, transform it, and load it into the central data repository.

• **Extract:** Extract data from the different data sources using Python libraries such as pandas, SQLite, or requests for reading data from various file formats, APIs, or databases.

• **Transform:** Apply data transformations using Python data manipulation techniques such as data cleaning, data validation, data enrichment, or data aggregation using pandas, NumPy, or other relevant libraries.

• **Load:** Load the transformed data into the central data repository such as a relational database, a data lake, or a data warehouse using Python libraries for database connectivity, such as SQLAlchemy or psycopg2.

3. **Data Quality and Error Handling**: Implement data quality checks and error handling mechanisms to ensure the integrity and accuracy of the data. For example, validate data types, check for duplicate records, handle null values, and log error records to error tables or files for further analysis and troubleshooting.

4. **Performance Optimization**: Optimize the performance of the data integration workflows by using Python techniques such as parallel processing, memory management, or data streaming to improve data processing speed and efficiency.

5. **Documentation:** Create documentation for the data integration workflows, including design specifications, mapping documents, and user guides, to facilitate future maintenance and support.

**Deliverables:**

• Data integration workflows in Python that extract, transform, and load data from the data sources into the central data repository.

• Error handling mechanisms and data quality checks to ensure the integrity of the data.

• Performance optimization techniques applied to improve data processing performance.

• Documentation including design specifications, mapping documents, and user guides.

This project will allow you to practice various Python concepts such as data extraction, transformation, and loading, error handling, data quality checks, performance optimization, and documentation. You can implement the required workflows, apply data transformations, and validate the data in the central data repository to gain hands-on experience with Python for data integration and improve your knowledge and skills. Good luck!