Overview

This documentation covers the set of PySpark scripts developed for the purpose of extracting, transforming, and loading (ETL) social media data into a structured format suitable for analysis. The process is split into distinct stages, each handled by a specific script:

1. **Extraction** - `Write_csv.py`
2. **Transformation** - `Cleaning.py`
3. **Validation and Logging** - `Logging.py`
4. **Loading** - `Connect_Dwh.py`

1. Extraction (Write_csv.py)

Objective: Convert complex JSON structures into a flat CSV format that simplifies further processing.

Description:

- Reads JSON from a specified file path.
- Flattens nested JSON fields (e.g., user data, posts, reactions, comments).
- Outputs a CSV file containing the flattened data.

Key Functions:

- `explode()`: Utilized to handle array structures within the JSON, breaking them into separate rows for each element.
- `withColumn()`: Used to manipulate and transform data types and structures.

2. Transformation (Cleaning.py)

Objective: Clean and preprocess the extracted CSV data to ensure quality and consistency.

Description:

- Converts string timestamps to actual timestamp data types.
- Casts numeric fields from strings to integers.
- Handles missing values by either removing or filling them based on the field's importance.

Key Functions:

- `cast()`: Converts data types of DataFrame columns.
- `fillna()`: Fills null values with predefined data.
- `dropDuplicates()`: Removes duplicate rows based on certain conditions.

3. Validation and Logging (Logging.py)

Objective: Validate data quality and log any discrepancies or errors found during validation.

Description:

- Checks for null values in critical columns.
- Validates age ranges to ensure data accuracy.
- Verifies email formats using regular expressions.
- Logs errors and informational messages regarding the validation process.

Key Functions:

- `filter()`: Filters DataFrame based on specified conditions.
- `rlike()`: Regex matching function to validate strings.

4. Loading (Connect_Dwh.py)

Objective: Load the cleaned and validated data into a PostgreSQL data warehouse.

Description:

- Establishes a connection to the PostgreSQL database using JDBC.
- Writes data into dimension and fact tables.
- Manages data partitioning to optimize the write process.

Key Functions:

- `write.format()`: Configures the format (JDBC) for the output data.
- `monotonically_increasing_id()`: Generates unique IDs for surrogate key fields.

Best Practices and Guidelines

- **PEP 8:** All scripts adhere to the PEP 8 style guide for Python code, ensuring readability and maintainability.
- **Logging:** Systematic logging is implemented to trace the ETL process and capture any unexpected behavior.
- **Error Handling:** Comprehensive error and exception handling strategies are in place to manage and resolve runtime errors.

Conclusion

These PySpark scripts form a robust framework for ETL processes, capable of handling large volumes of data efficiently while maintaining data integrity and accuracy. Each script is designed to be modular and reusable, allowing for adjustments and enhancements without disrupting the overall workflow.