Data Validation Test Documentation

Test Results Summary

The validation tests were performed on all CSV files in the `files` directory. The tests focused on two main scenarios:

1. **Duplicate Columns Check**
    - All files passed the duplicate columns validation
    - No duplicate column names were found in any of the files
2. **Yes/No Fields Validation**
    - All files passed the Yes/No fields validation
    - All Yes/No type fields contain valid values ('Yes' or 'No')

Potential Issues and Limitations

1. **File Format Support**
    - Currently, the validator only supports CSV and JSON files
    - Other file formats will be skipped without validation
    - Recommendation: Add support for additional file formats if needed
2. **Data Type Validation**
    - The current implementation only validates Yes/No fields
    - Other data types (numbers, dates, etc.) are not validated
    - Recommendation: Extend validation to cover other data types based on requirements
3. **Performance Considerations**
    - Large files might cause memory issues as the entire file is loaded into memory
    - Recommendation: Implement chunked reading for large files
4. **Error Handling**
    - Basic error handling is implemented
    - Some edge cases might not be covered
    - Recommendation: Add more comprehensive error handling and logging

Non-Technical Explanation

Dear Manager/Stakeholder,

I've implemented a data validation system that helps ensure the quality and consistency of our data files. Here's what it does in simple terms:

1. **Duplicate Checker**
    - Think of this like checking for duplicate names in a list
    - It ensures that each column in our data has a unique name
    - This prevents confusion and data processing errors
2. **Yes/No Validator**
    - This is like checking that all checkboxes in a form are properly marked
    - It ensures that any field that should only contain "Yes" or "No" actually contains these values
    - This maintains data consistency and prevents invalid entries

The system generates a clear HTML report that shows:

- Which files were checked
- What tests were performed
- Whether each file passed or failed the tests
- Any specific issues found

This helps us maintain data quality and catch potential problems before they affect our business processes.

Future Improvements

1. **Additional Validations**
    - Add checks for required fields
    - Validate date formats
    - Check for data ranges and limits
2. **User Interface**
    - Create a web interface for running tests
    - Add ability to schedule automated tests
    - Implement email notifications for test results
3. **Reporting**
    - Add trend analysis over time
    - Generate PDF reports
    - Create dashboards for test results

How to Use

1. Place your data files in the `files` directory
2. Run the test script:

```
python3 SendtoCandidate/tests/test_validator.py
```

3. Check the generated report in the `evidence` folder
4. Review any issues found and take necessary actions

Technical Requirements

- Python 3.x
- Required Python packages:
  - pandas
  - openpyxl (for Excel file support)