IRisk Lab Data Discovery and Consolidation - Task 2
Week 3 Report
Sep. 13
Rohit Valmeekam

<u>What was accomplished</u>

Created basic data pipeline and tested it on discovered data , including the following steps :

- **Input Data Collection:**
  - The process begins with a Luigi task called ConsolidateData.
  - This task is responsible for collecting input data from a folder named input_data.
- **Input Data Parsing:**
  - Within the ConsolidateData task, Luigi identifies all files in the input_data folder with a .csv extension.
  - For each of these input files, a Luigi sub-task of type ReadCSV is created.
  - The ReadCSV task is responsible for reading the content of each input CSV file.
- **Dataframe Creation:**
  - Inside the ReadCSV task, the input CSV file is read using pandas, creating a dataframe containing the data.
- **Intermediary CSV Creation:**
  - After reading the input CSV file, the data is written to an intermediary CSV file named "intermediate.csv".
  - This intermediary CSV file serves as a temporary storage for the data before further processing within the ConsolidateData task.
- **Data Processing Algorithm:**
  - Back in the ConsolidateData task, the intermediary CSV file is processed using a data processing algorithm.
  - This algorithm includes several steps, such as categorization, folder creation, missing value removal, and output file placement.
- **Category Assignment:**
  - For each input file, the algorithm prompts the user to specify the category to which the dataset belongs.
  - This user input determines the subfolder where the processed data will be saved.
- **Folder Creation:**

- ➢ If the category-specific folder does not exist within the output directory, it is created.
- ➢ This ensures that datasets of the same category are stored together in their respective folders.

❖ **Missing Value Removal:**

- ➢ Before saving the processed dataset, any rows with missing values (NaN) are removed from the dataframe.
- ➢ This step ensures that only complete data is included in the output.

❖ **Data Output:**

- ➢ The processed data frame is then saved as a new CSV file within the appropriate category folder.
- ➢ The path to this output file includes both the output directory and the category-specific subfolder.

❖ **Temporary Database (Optional):**

- ➢ The cleaned dataset is then read into a dummy database created in SQLLite
- ➢ The input file is then deleted in from the input folder

❖ **Input File Deletion:**

- ➢ After successfully processing an input CSV file, it is deleted from the input_data folder.
- ➢ This ensures that only unprocessed files remain in the input_data folder.

❖ **Check for Remaining Files:**

- ➢ Within the ConsolidateData task, there is a complete() function.
  - ■ This function checks if there are any remaining CSV files in the input_data folder.
  - ■ If there are still unprocessed files, the algorithm continues to run and process them.

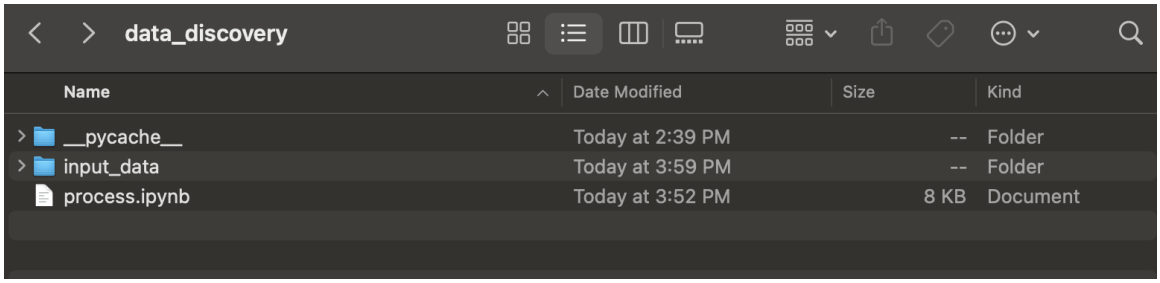
This process is designed to systematically process a collection of CSV files, categorize them, clean the data, and organize the results into category-specific folders within the output directory. It also maintains the input_data folder by deleting processed files and checking for any remaining unprocessed ones.

<u>Challenges</u>

❖ Deciding the technologies necessary for the creation of the Python Script
❖ Structuring the data pipeline
❖ Deciding what processes the processing algorithm should cover
❖ Connecting the data pipeline to the database

Next Steps

- **Handle Non-CSV Files:**
  - Modify the script to handle files with extensions other than .csv.
  - Update the algorithm to determine the file format based on the extension.
  - Implement logic to read and process data from non-CSV files accordingly.
  - If additional processing steps are required for specific file formats, add them to the processing script.
- **Replace Dummy SQL Database:**
  - Replace the dummy SQLite database with the final database that you intend to use.
  - Ensure that the database schema is aligned with the structure of the output data.
  - Establish a connection to the final database within the script.
  - Modify the script to insert or update data in the final database as needed, based on the processed data.
- **Database Schema Design:**
  - Design the schema of the final database to accommodate the data you are processing.
  - Define tables, columns, and relationships to organize and store the data effectively.
  - Consider data types, constraints, and indexing for optimal database performance.



Example of the file structure before processing

After processing will be displayed during meeting and added to report after meting