1. Modules and Libraries

- **pandas:** Used for data manipulation and analysis.
- **google.oauth2.service_account:** Handles authentication using Google service account credentials.
- **googleapiclient.discovery:** Accesses the Google Sheets API.
- **googleapiclient.errors.HttpError:** Handles API-related errors.
- **dotenv.load_dotenv:** Loads environment variables from a .env file.

---

2. GoogleSheetUtils Class

This class provides utility functions to interact with Google Sheets.

a. load_credentials

- **Purpose:** Load Google service account credentials from a file.
- **Checks:** Validates if the file exists.
- **Returns:** An authenticated credentials object.

b. build_service

- **Purpose:** Create a Google Sheets API service object.
- **Input:** Credentials from load_credentials.
- **Output:** A Sheets API service object.

c. fetch_sheet_data

- **Purpose:** Retrieve data from a specific tab and range in a Google Sheet.
- **Input:**
  - service: The Sheets API service object.
  - spreadsheet_id: The ID of the spreadsheet.
  - tab_name and range_: Specify the tab and range to fetch.
- **Output:** The retrieved data as a list of lists.

d. update_sheet_with_dataframe

- **Purpose:** Update a Google Sheet with data from a pandas DataFrame.
- **Steps:**
  1. Convert the DataFrame to a list of lists.
  2. Define the range (starting cell).
  3. Use sheet.values().update to write data to the sheet.

- **Error Handling:** Catches API errors and prints the error message.

---

3. DataFrameUtils Class

This class provides functions for pandas DataFrame manipulation.

a. process_data_to_dataframe

- **Purpose:** Convert raw Google Sheets data to a structured pandas DataFrame.
- **Steps:**
    1. First row of data becomes the header.
    2. Remaining rows become data entries.
    3. Missing values are filled with "N/A".

b. filter_dataframe

- **Purpose:** Filter DataFrame rows by a column value and select a subset of columns.
- **Input:** Column name, filter value, start and end column indices.

c. merge_columns

- **Purpose:** Merge duplicate columns by concatenating their values.
- **Steps:**
    1. Combine the specified columns into a single column.
    2. Drop redundant columns.

d. handle_trip_ids

- **Purpose:** Expand rows with multiple trip IDs into separate rows.
- **Steps:**
    1. Split trip_column values by commas.
    2. Create a new row for each trip ID.

e. match_trip_details

- **Purpose:** Match trip details from another DataFrame.
- **Steps:**
    1. Strip whitespaces from column names.
    2. Match rows in one DataFrame based on the trip_column value in another.

f. add_cn_number

- **Purpose:** Assign sequential credit note (CN) numbers to each row.
- **Input:** Starting CN number and column name.

g. update_status_and_link

- **Purpose:** Update the status and add hyperlinks for saved PDF credit notes.
- **Steps:**
    1. If a valid PDF path exists, set status_column to "Saved" and add a hyperlink.
    2. Otherwise, mark status as "Not Saved".

How the Code Could Be Used

1. **Authentication:** Use GoogleSheetUtils.load_credentials to authenticate with Google Sheets.
2. **Data Retrieval:** Use fetch_sheet_data to pull data from a Google Sheet.
3. **Data Processing:** Use DataFrameUtils methods to clean, filter, or transform the data.
4. **Sheet Update:** Write the processed DataFrame back to the sheet with update_sheet_with_dataframe.

This modular design ensures clean separation of concerns, making it reusable and easy to test.