**Game Data Processing**

This script processes game data from multiple CSV files stored in separate folders for each game. It combines these files into a single cohesive DataFrame and generates a summary of each game's data. The script ensures the integrity of the data by checking for required files, processing various columns, and adjusting formats when necessary. The final output consists of individual CSV files for each game and a comprehensive df\_steamgame.csv that contains data for all the games.

**Key Functions**

* **Folder and File Structure**: The script loops through the ../data/raw/game lib folder, identifying subfolders for each game. It expects each folder to contain the following required CSV files:
  + overview.csv
  + copies sold.csv
  + revenue.csv
  + top seller rank.csv
  + ccu.csv
  + price.csv
  + review.csv
  + followers.csv
  + average playtime.csv
  + outstanding wishlist.csv
  + positive reviews.csv
* **File Validation**: It checks that each game folder contains all required CSV files. If all files are present, the game data is processed.
* **Data Processing**: The script processes each of the CSV files into a consistent format, handling date and value columns, filling gaps in the price.csv when necessary. It also extracts and categorizes tags from the overview.csv, splitting them into predefined categories such as 'game type', 'style', 'mechanics', and 'perspective'.
* **df\_price Specific Handling**: The price.csv can have two different formats. In some cases, it contains only two columns (date and price), which require special processing to fill in missing dates. In other cases, it follows the same structure as the smaller data files (revenue, outstanding wishlist, positive reviews, etc.), making it easier to process similarly to other data frames.
* **Daily Changes**: New columns are added to reflect daily changes in copies sold, revenue, reviews, and outstanding wishlist.
* **Final Output**: After processing the data, individual gameinfo.csv files are saved for each game. All game data is then consolidated into a massive df\_steamgame.csv, which contains the combined data for all games.

**Summary Files**

* **game\_info\_summary.csv**: Contains basic metadata for each game, including the folder location and processed file paths.
* **df\_steamgame.csv**: A comprehensive CSV containing all processed game data.

**Price Data**

* **df\_price Processing**: The script handles two different formats of price data. In one format, the file contains just two columns for date and price, which requires interpolation to fill in missing dates. In the other format, it matches the structure of other small data frames, making it simpler to process. Both formats are handled appropriately to ensure consistency across the dataset.

import os

import pandas as pd

def process\_game\_data(main\_folder):

    # List of required files

    required\_files = [

        'overview.csv', 'copies sold.csv', 'revenue.csv', 'top seller rank.csv',

        'ccu.csv', 'price.csv', 'review.csv', 'followers.csv',

        'average playtime.csv', 'outstanding wishlist.csv', 'positive reviews.csv'

    ]

    # Initialize a DataFrame to store game info

    df\_info = pd.DataFrame(columns=['game\_name', 'location'])

    # Iterate through each subfolder in the main folder

    for game\_folder in os.listdir(main\_folder):

        game\_path = os.path.join(main\_folder, game\_folder)

        if os.path.isdir(game\_path):

            # Check if all required files are present

            if all(os.path.isfile(os.path.join(game\_path, file)) for file in required\_files):

                # Load each file into a DataFrame

                df\_overview = pd.read\_csv(os.path.join(game\_path, 'overview.csv'))

                df\_copies\_sold = pd.read\_csv(os.path.join(game\_path, 'copies sold.csv'))

                df\_revenue = pd.read\_csv(os.path.join(game\_path, 'revenue.csv'))

                df\_top\_seller\_rank = pd.read\_csv(os.path.join(game\_path, 'top seller rank.csv'))

                df\_ccu = pd.read\_csv(os.path.join(game\_path, 'ccu.csv'))

                df\_price = pd.read\_csv(os.path.join(game\_path, 'price.csv'))

                df\_reviews = pd.read\_csv(os.path.join(game\_path, 'review.csv'))

                df\_followers = pd.read\_csv(os.path.join(game\_path, 'followers.csv'))

                df\_average\_playtime = pd.read\_csv(os.path.join(game\_path, 'average playtime.csv'))

                df\_outstanding\_wishlist = pd.read\_csv(os.path.join(game\_path, 'outstanding wishlist.csv'))

                df\_positive\_reviews = pd.read\_csv(os.path.join(game\_path, 'positive reviews.csv'))

                # Process df\_price based on the number of columns

                if len(df\_price.columns) > 2:

                    df\_price = df\_price.T

                    df\_price = df\_price[1:]

                    df\_price = df\_price.reset\_index()

                    df\_price = df\_price.rename(columns={'index': 'date'})

                    df\_price['date'] = df\_price['date'].str[4:]

                    df\_price = df\_price.rename(columns={0: 'price'})

                else:

                    df\_price = df\_price.rename(columns={'x': 'date', 'y': 'price'})

                    df\_price['date'] = pd.to\_datetime(df\_price['date'], unit='ms')

                    filled\_rows = []

                    for i in range(len(df\_price) - 1):

                        current\_price = df\_price.loc[i, 'price']

                        current\_date = df\_price.loc[i, 'date']

                        next\_date = df\_price.loc[i + 1, 'date']

                        date\_range = pd.date\_range(start=current\_date, end=next\_date, freq='D')

                        for date in date\_range:

                            filled\_rows.append({'date': date, 'price': current\_price})

                    filled\_rows.append({'date': df\_price.loc[len(df\_price) - 1, 'date'], 'price': df\_price.loc[len(df\_price) - 1, 'price']})

                    df\_price = pd.DataFrame(filled\_rows)

                    df\_price.reset\_index(drop=True, inplace=True)

                    df\_price['date'] = df\_price['date'].dt.normalize()

                    df\_price = df\_price.drop\_duplicates(subset='date', keep='first').reset\_index(drop=True)

                    df\_price = df\_price.drop(df\_price.index[-1])

                # Process other DataFrames

                def process\_df(df, column\_name):

                    df = df.T

                    df = df[1:]

                    df = df.reset\_index()

                    df = df.rename(columns={'index': 'date'})

                    df['date'] = df['date'].str[4:]

                    df = df.rename(columns={0: column\_name})

                    return df

                df\_copies\_sold = process\_df(df\_copies\_sold, 'copies sold')

                df\_revenue = process\_df(df\_revenue, 'revenue')

                df\_top\_seller\_rank = process\_df(df\_top\_seller\_rank, 'top seller rank')

                df\_ccu = process\_df(df\_ccu, 'ccu')

                df\_reviews = process\_df(df\_reviews, 'reviews')

                df\_followers = process\_df(df\_followers, 'followers')

                df\_average\_playtime = process\_df(df\_average\_playtime, 'average playtime')

                df\_outstanding\_wishlist = process\_df(df\_outstanding\_wishlist, 'outstanding wishlist')

                df\_positive\_reviews = process\_df(df\_positive\_reviews, 'positive reviews')

                # Process df\_overview

                tag\_categories = {

                    'tag game type': [

                        'Action Roguelike', '2D Platformer', 'Hack and Slash', 'Roguelite', 'Action RPG', 'FPS', 'Multiplayer',

                        'Open World', 'Survival', 'Crafting', 'Co-op Campaign', 'Base Building', 'Simulation', 'Exploration',

                        'Visual Novel', 'Shooter', 'RPG', 'Strategy', 'Sandbox', 'Action-Adventure', 'Sports', 'Puzzle', 'Fighting',

                        'Roguelike', 'Adventure', 'Card Game', 'RTS'

                    ],

                    'tag style': [

                        'Souls-like', 'Cyberpunk', 'Story Rich', 'Multiple Endings', 'Anime', 'Pixel Graphics', 'Retro', 'Cartoony',

                        'Fantasy', 'Dark Fantasy', 'Comedy', 'LGBTQ+', 'Realistic', 'Gothic', 'Mythology', 'Noir', 'Dystopian',

                        'Sci-fi', 'Lovecraftian', 'Atmospheric', 'Cute'

                    ],

                    'tag mechanics': [

                        'Side Scroller', 'Controller', 'Choices Matter', 'Crafting', 'Base Building', 'Co-op Campaign',

                        'Turn-Based', 'Real-Time', 'Procedural Generation', 'Deckbuilding', 'Inventory Management',

                        'Physics', 'Roguelike Deckbuilder', 'Tactical', 'PvP', 'PvE', 'Resource Management', 'Exploration', 'Combat',

                        'Building', 'Moddable', 'Level Editor', 'Narration'

                    ],

                    'tag perspective': [

                        '2D', '3D', 'Isometric', 'First-Person', 'Third Person', 'Top-Down', 'Side-Scrolling', 'Third-Person Shooter',

                        'VR', 'Over-the-Shoulder', '3D Vision', '2.5D', 'First-Person Shooter'

                    ]

                }

                def categorize\_tags(tag\_string, categories):

                    tags = tag\_string.split(',')

                    categorized = {key: [] for key in categories}

                    for tag in tags:

                        tag = tag.strip()

                        found = False

                        for category, keywords in categories.items():

                            if tag in keywords:

                                categorized[category].append(tag)

                                found = True

                        if not found:

                            continue

                    for category in categorized:

                        if not categorized[category]:

                            categorized[category] = ['-']

                    return categorized

                for category in tag\_categories.keys():

                    df\_overview[category] = df\_overview['tags'].apply(lambda x: ', '.join(categorize\_tags(x, tag\_categories)[category]) if pd.notnull(x) else '-')

                df\_overview = df\_overview.drop(columns=['tags'])

                df\_overview = df\_overview.drop(columns=['languages', 'price', 'features', 'early access exit date', 'description', 'first release date'])

                genres\_split = df\_overview['genres'].str.split(',', expand=True)

                df\_overview = df\_overview.assign(

                    genre\_1=genres\_split.get(0, '-'),

                    genre\_2=genres\_split.get(1, '-'),

                    genre\_3=genres\_split.get(2, '-')

                )

                cols = df\_overview.columns.tolist()

                dev\_idx = cols.index('developers')

                new\_cols\_order = cols[:dev\_idx+1] + ['genre\_1', 'genre\_2', 'genre\_3'] + cols[dev\_idx+1:]

                df\_overview = df\_overview[new\_cols\_order]

                df\_overview = df\_overview.drop(columns=['genres'])

                df\_overview\_repeated = pd.concat([df\_overview]\*len(df\_price), ignore\_index=True)

                df\_game = pd.concat([df\_overview\_repeated.reset\_index(drop=True), df\_price.reset\_index(drop=True)], axis=1)

                df\_game = pd.concat([df\_game, df\_copies\_sold.reset\_index(drop=True)], axis=1)

                df\_game = pd.concat([df\_game, df\_revenue.reset\_index(drop=True)], axis=1)

                df\_game = pd.concat([df\_game, df\_top\_seller\_rank.reset\_index(drop=True)], axis=1)

                df\_game = pd.concat([df\_game, df\_ccu.reset\_index(drop=True)], axis=1)

                df\_game = pd.concat([df\_game, df\_reviews.reset\_index(drop=True)], axis=1)

                df\_game = pd.concat([df\_game, df\_followers.reset\_index(drop=True)], axis=1)

                df\_game = pd.concat([df\_game, df\_average\_playtime.reset\_index(drop=True)], axis=1)

                df\_game = pd.concat([df\_game, df\_outstanding\_wishlist.reset\_index(drop=True)], axis=1)

                df\_game = pd.concat([df\_game, df\_positive\_reviews.reset\_index(drop=True)], axis=1)

                df\_game = df\_game.loc[:, ~df\_game.columns.duplicated(keep='first')]

                is\_free\_game = df\_price['price'].eq(0).all()

                df\_game['free game'] = is\_free\_game

                if not is\_free\_game:

                    non\_zero\_price\_index = df\_price[df\_price['price'] != 0].index

                    df\_game = df\_game.loc[non\_zero\_price\_index].reset\_index(drop=True)

                    df\_price = df\_price.loc[non\_zero\_price\_index].reset\_index(drop=True)

                df\_game['daily copies sold'] = df\_game['copies sold'].diff().fillna(0)

                df\_game['daily revenue'] = df\_game['revenue'].diff().fillna(0)

                df\_game['daily reviews'] = df\_game['reviews'].diff().fillna(0)

                df\_game['daily outstanding wishlist'] = df\_game['outstanding wishlist'].diff().fillna(0)

                gameinfo\_path = os.path.join(game\_path, 'gameinfo.csv')

                df\_game.to\_csv(gameinfo\_path, index=False)

                print(f"Game info for {game\_folder} saved at {gameinfo\_path}")

                df\_info = pd.concat([df\_info, pd.DataFrame({'game\_name': [game\_folder], 'location': [gameinfo\_path]})], ignore\_index=True)

    df\_info.to\_csv(os.path.join(main\_folder, 'game\_info\_summary.csv'), index=False)

    # Combine all gameinfo.csv files into a massive DataFrame

    df\_steamgame = pd.DataFrame()

    for game\_folder in os.listdir(main\_folder):

        game\_path = os.path.join(main\_folder, game\_folder)

        gameinfo\_path = os.path.join(game\_path, 'gameinfo.csv')

        if os.path.isfile(gameinfo\_path):

            df\_game = pd.read\_csv(gameinfo\_path)

            df\_steamgame = pd.concat([df\_steamgame, df\_game], ignore\_index=True)

    df\_steamgame.to\_csv(os.path.join(main\_folder, 'df\_steamgame.csv'), index=False)

    print("ALL TASKS DONE!!!!!!")

# Example usage

main\_folder = '../data/raw/game lib'

process\_game\_data(main\_folder)