İmport pandas as np

# ('../data/raw/test field') is our main directory

**Cell1**

#we will create our first game lib dataframe from HellDivers 2 game

df\_copies\_sold = pd.read\_csv('../data/raw/test field/Anomaly Agent/copies sold.csv')

df\_average\_playtime = pd.read\_csv('../data/raw/test field/Anomaly Agent/average playtime.csv')

df\_revenue = pd.read\_csv('../data/raw/test field/Anomaly Agent/revenue.csv')

df\_reviews = pd.read\_csv('../data/raw/test field/Anomaly Agent/review.csv')

df\_ccu = pd.read\_csv('../data/raw/test field/Anomaly Agent/ccu.csv')

df\_followers = pd.read\_csv('../data/raw/test field/Anomaly Agent/followers.csv')

df\_outstanding\_wishlist = pd.read\_csv('../data/raw/test field/Anomaly Agent/outstanding wishlist.csv')

df\_positive\_reviews = pd.read\_csv('../data/raw/test field/Anomaly Agent/positive reviews.csv')

df\_top\_seller\_rank = pd.read\_csv('../data/raw/test field/Anomaly Agent/top seller rank.csv')

df\_price = pd.read\_csv('../data/raw/test field/Anomaly Agent/price.csv')

#---------------------------------Data Clearing---------------------------------#

df\_overview = pd.read\_csv('../data/raw/test field/Anomaly Agent/overview.csv')

We loaded all files as data frames. Now we will first setup df\_price file for beginning, because it has different format than other 9 files (also excluding overview file).

**Cell2**

# in df\_price df we change x column to date and y column to price

#and we dont need to rotate the dataframe because it is already in the right format

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

# Convert 'date' column to datetime format (assume date is in milliseconds)

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

# Create a list to store the filled rows

filled\_rows = []

# Iterate through the DataFrame, find price changes, and fill missing dates

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']

    # Create a date range between the current date and the next date

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

    # For each date in the range, append the date and current price to the filled rows list

    for date in date\_range:

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

# Add the last row of the original DataFrame (last price point)

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

# Update df\_price with the filled data

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

# Reset index to ensure it starts from 0

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

# Ensure that all date entries are set to 00:00:00 by normalizing the dates

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

#to find repetating dates only under date column and remove them

#because we only need one price for each date

#reset index

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

#and drop the last row

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

Column building setup. From this point we will convert the list of files that will shaped for our final data frame:

copies sold

revenue

top seller rank

CCU

Review

Followers

average playtime

outstanding wishlist

positive reviews

Each of these files will be columns of our new data frame, and by the help of these files we will also add some more columns.

**Adding each df 1 by 1 without using a for loop or smiliar method. When we start automazion we will create a for loop for these 9 files and program will convert and tranpose each one and get ready to be added to dataframe**

**Cell3**

copies\_sold.csv file

#we transpose the dataframe to make the date as the index

#then we make the first row as the columns

#then we reset the index

#then we rename the index to date

df\_copies\_sold = df\_copies\_sold.T

df\_copies\_sold = df\_copies\_sold[1:]

df\_copies\_sold = df\_copies\_sold.reset\_index()

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

#drop weekdays from date column that indicated by first 3 characters

df\_copies\_sold['date'] = df\_copies\_sold['date'].str[4:]

#name 0 column as copies sold

df\_copies\_sold = df\_copies\_sold.rename(columns = {0:'copies sold'})

**Cell 4:**

revenue.csv

df\_revenue = df\_revenue.T

df\_revenue = df\_revenue[1:]

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

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

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

df\_revenue = df\_revenue.rename(columns = {0:'revenue'})

**Cell 5:**

top\_seller\_rank.csv

df\_top\_seller\_rank = df\_top\_seller\_rank.T

df\_top\_seller\_rank = df\_top\_seller\_rank[1:]

df\_top\_seller\_rank = df\_top\_seller\_rank.reset\_index()

df\_top\_seller\_rank = df\_top\_seller\_rank.rename(columns = {'index':'date'})

df\_top\_seller\_rank['date'] = df\_top\_seller\_rank['date'].str[4:]

df\_top\_seller\_rank = df\_top\_seller\_rank.rename(columns = {0:'top seller rank'})

df\_followers = df\_followers.rename(columns={df\_followers.columns[1]: 'top seller rank'})

**Cell 6:**

ccu.csv

df\_ccu = df\_ccu.T

df\_ccu = df\_ccu[1:]

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

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

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

df\_ccu = df\_ccu.rename(columns = {0:'ccu'})

**Cell 7:**

reviev.csv

df\_reviews = df\_reviews.T

df\_reviews = df\_reviews[1:]

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

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

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

df\_reviews = df\_reviews.rename(columns = {0:'reviews'})

**Cell 8:**

followers.csv

df\_followers = df\_followers.T

df\_followers = df\_followers[1:]

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

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

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

df\_followers = df\_followers.rename(columns = {0:'followers'})

df\_followers = df\_followers.rename(columns={df\_followers.columns[1]: 'followers'})

**Cell 9:**

average\_playtime.csv

df\_average\_playtime = df\_average\_playtime.T

df\_average\_playtime = df\_average\_playtime[1:]

df\_average\_playtime = df\_average\_playtime.reset\_index()

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

df\_average\_playtime['date'] = df\_average\_playtime['date'].str[4:]

df\_average\_playtime = df\_average\_playtime.rename(columns = {0:'average playtime'})

**Cell 10:**

outstanding\_wishlist.csv

df\_outstanding\_wishlist = df\_outstanding\_wishlist.T

df\_outstanding\_wishlist = df\_outstanding\_wishlist[1:]

df\_outstanding\_wishlist = df\_outstanding\_wishlist.reset\_index()

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

df\_outstanding\_wishlist['date'] = df\_outstanding\_wishlist['date'].str[4:]

df\_outstanding\_wishlist = df\_outstanding\_wishlist.rename(columns = {0:'outstanding wishlist'})

**Cell 11:**

positive\_reviews.csv

df\_positive\_reviews = df\_positive\_reviews.T

df\_positive\_reviews = df\_positive\_reviews[1:]

df\_positive\_reviews = df\_positive\_reviews.reset\_index()

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

df\_positive\_reviews['date'] = df\_positive\_reviews['date'].str[4:]

df\_positive\_reviews = df\_positive\_reviews.rename(columns = {0:'positive reviews'})

**Root structure of cell 3 to cell 11:**

**Transposes the DataFrame, switching rows with columns**

df\_ = df\_.T

**Removes the first row (likely metadata or unnecessary information)**

df\_ = df\_[1:]

**Resets the index, turning the current index into a column, and creating a new default index**

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

**Renames the newly created "index" column to "date"**

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

**Slices each date string, removing the first 4 characters from the 'date' column**

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

**Renames the first column (formerly the 0th column) to "revenue"**

df\_ = df\_.rename(columns = {0:'df\_(name)'})

**Pre-processing overview:**

**Overview data frale there is a tag column indicates game tags. It may have massive information inside of it.**

**With this code we will seperate tags, and place them under representative column**

becore this piece of code, in a different notebook file, I found necessary tags from all the game data frames that I have,

collected them into a list and then I created a new dataframe from that list

under these lines, I identified all tag\_categories and tags for each game by help of AI.

**Cell 12:**

# Define the tag categories with the new format

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'

    ]

}

# Function to assign tags to categories with a single '-' if no match found

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

    tags = tag\_string.split(',')  # Split tags by commas

    categorized = {key: [] for key in categories}  # Prepare empty lists for each category

    for tag in tags:

        tag = tag.strip()  # Clean the tag

        found = False

        for category, keywords in categories.items():

            if tag in keywords:

                categorized[category].append(tag)

                found = True

        if not found:

            continue

    # Fill empty categories with a single '-'

    for category in categorized:

        if not categorized[category]:

            categorized[category] = ['-']  # Use single '-' if no tag matches

    return categorized

# Apply the function to df\_overview and create the new columns

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 '-')

# Drop the original 'tags' column

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

**Adding Genre columns to overview dataframe (Genre 1, Genre 2, Genre 3), after that combining desigrnated dataframes as new columns**

**Cell 13:**

# Step 1: Remove 'price' column from df\_overview, since we already have it in df\_price

#and also removing languages, features, early access exit date, description columns

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

#------------------------------------

# Split the 'genres' column into three new columns ('genre 1', 'genre 2', 'genre 3')

df\_overview[['genre 1', 'genre 2', 'genre 3']] = df\_overview['genres'].str.split(',', expand=True)[[0, 1, 2]]

# Move the new 'genre 1', 'genre 2', 'genre 3' columns next to 'developers' column

# Get the current column list

cols = df\_overview.columns.tolist()

# Find the position of 'developers' column

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

# Rearrange columns: developers + genre 1, 2, 3 next to developers

new\_cols\_order = cols[:dev\_idx+1] + ['genre 1', 'genre 2', 'genre 3'] + cols[dev\_idx+1:]

# Reorder the dataframe columns

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

# Drop the original 'genres' column

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

#------------------------------------

# Step 2: Repeat df\_overview to match the length of df\_price

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

# Step 3: Concatenate df\_overview\_repeated with df\_price

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

# Step 4: Repeat the same logic for the other dataframes: df\_copies\_sold, df\_revenue, df\_top\_seller\_rank, df\_ccu, df\_review, df\_followers, df\_average\_playtime, df\_outstanding\_wishlist, df\_positive\_reviews

# Concatenate all other dataframes one by one

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)

# Now df\_game contains df\_overview, df\_price, and all the other dataframes combined

# Find all columns with the name 'date'

date\_columns = [col for col in df\_game.columns if col == 'date']

# Keep only the first occurrence of 'date' and remove the rest

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

**Identifying if the game is free or not. Adding a boolean column ‘free game’ with True and False indicator.**

**If game is not free False) it will find all the 0 prices from data frame and remove all 0 priced rows. This indicates when the game started to be sold, which will be our starting point for game data frame.**

**Cell 14:**

# Step 1: Check if the game is free by verifying if all price values in df\_price are 0.

# If all prices are 0, the game is free, otherwise, it's not.

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

# Step 2: Add a new column 'free game' to df\_game, where all rows get the same value based on whether the game is free.

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

# Step 3: If the game is not free, check for rows with price 0 and remove those rows.

#by this filter, we can know when the game is not free

#and if the game is not free, we can itentify when the game first starts to be sold

if not is\_free\_game:

    # Filter out rows from df\_game and df\_price where price is 0.

    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)

**Last implementation on data frame.**

**There are 4 columns that indicates cumulative outcome of ‘copies sold’, ‘revenue’, ‘reviews’ and ‘outstanding wishlist’. I also create a Daily change of these columns adding Daily tag front of each column and create them. Calculation of each information is simple; current row substracts previous row information and add the volume to daily column.**

**Cell 15:**

# Calculate daily changes for 'copies sold', 'revenue', 'reviews' and ‘outstanding wishlist’

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)

# Display the updated dataframe

df\_game.head()

**Explanation of this data:**

**Cell 1**:  
We first import the required data from CSV files related to a game called *Anomaly Agent*. These include various game statistics like copies sold, revenue, playtime, reviews, etc. Each CSV is loaded into a Pandas DataFrame.

**Cell 2**:  
We start processing the df\_price DataFrame, which contains price data in a slightly different format compared to other data files. We rename the columns to date and price, convert the date format, and then fill in missing dates by creating daily entries for the price.

* **Steps within Cell 2**:
  1. Rename columns x to date and y to price.
  2. Convert date from milliseconds to human-readable format.
  3. Ensure that for each day between two recorded price points, the price is copied forward to fill any gaps.
  4. Remove duplicate dates, keeping only the first occurrence, and drop the last row (to avoid any outlier or incomplete data).

**Cell 3-11**:  
Each of these cells follows a similar process to handle the other game-related files (copies sold, revenue, rank, etc.).

* **Common Steps**:
  1. Transpose the DataFrame, switching rows with columns to make dates as rows.
  2. Clean and reset the index to ensure that date becomes the main column.
  3. Drop unnecessary characters from the date column (first four characters).
  4. Rename the first data column to match the file’s content (e.g., copies sold, revenue, etc.).

**Cell 12**:  
This cell handles the processing of tags, which are pieces of metadata that describe game attributes (like genres, mechanics, perspectives, etc.). A function is used to assign each tag to a category, which is then split into four columns:

* tag game type
* tag style
* tag mechanics
* tag perspective

If no relevant tag is found for a category, a single '-' is assigned to represent missing data. After processing, the original 'tags' column is removed from the DataFrame.

**Cell 13**:  
This step focuses on cleaning up the df\_overview DataFrame and ensuring proper column alignment:

* Removing redundant columns like price and languages.
* Splitting the genres column into genre 1, genre 2, and genre 3.
* Reordering columns to make the DataFrame more organized, placing genre-related information next to developers.
* The df\_overview is repeated to match the length of the df\_price DataFrame so that it can be merged seamlessly later.

Finally, all cleaned and processed DataFrames are concatenated together, ensuring that the information from various sources (price, revenue, reviews, etc.) is aligned on the same date column, creating the final df\_game DataFrame with all relevant game information.