CS2316 HOMEWORK 07: STEAM STORE ANALYZER

Purpose

This homework will give you practice using two very important Python modules for data manipulation: NumPy and Pandas. Numpy is a powerful library for performing mathematical and logical operations on Arrays. It provides an abundance of useful features for operations on n-arrays and matrices in Python. Pandas builds upon NumPy adding even more features and the ability to handle tabular data from nonhomogeneous types. Both of these modules are commonly used by data scientists and analysts.

Skills

After learning Numpy and Pandas, you'll be able to input, clean, and aggregate large quantities of data, and then use that data with other Python modules such as SciPy that is used for statistical analysis or Matplotlib that is used for visualizing the data.

Knowledge

Knowledge of these two very powerful modules will add to your overall Python and data manipulation skills. When given a new data set in your future career, you will have some powerful options on how to find answers to questions about the data.

Important

1. Due Date: **04/20/2020 at 11:59 pm**
2. This homework is graded out of **100** points, but you will have a possible **50** additional bonus points available.
3. This is an individual assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 2316, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission.
4. For Help:
    - TA Helpdesk (Schedule posted on class website)
    - Email TA's or use Piazza Forums Notes
    - How to Think Like a Computer Scientist
    - [http://openbookproject.net/thinkcs/python/english3e/]
    - [Numpy Reference Guide PDF], [Numpy User Guide PDF]
    - [https://pandas.pydata.org/docs/user_guide/io.html]
5. Comment out or delete all your function calls. Only global variables, and comments are okay to be outside the scope of a function. When your code is run, all it should do is run without any errors.

6. Do not wait until the last minute to do this assignment in case you run into problems.
7. **Read the entire specifications document before starting this assignment.**
8. **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%**

Introduction

The goal of this homework is to enhance your understanding of both the Numpy and Pandas libraries and how to utilize each. This homework will be separated into two sections. Part one will ask you to demonstrate your understanding of multidimensional array objects while creating functions that will manipulate and create arrays. Part two will consist of demonstrating your knowledge on using Pandas dataframes. You will be asked to define functions that will manipulate and create dataframes.

Background

You have just received an email from Valve Corporation, one of the most well-known video game developers and distributors in America, that they would like to set up an interview for an internship as a data analyst with you! In addition to the confirmation of an interview, the email also includes a file containing all of the statistics of games released on their Steam Store. Overjoyed with this news, you decide to start to create a subset of the dataset and start practicing your Numpy and Pandas skills for the technical part of the interview. Here is what you know about each field within the dataset:

- **appid:** a unique identifier for each title
- **name:** the name of the application/game
- **release_year:** the release year
- **english:** the language support of the game – 1 if English is supported, otherwise 0
- **developer:** name(s) of the developer(s). Separated by semicolons if there are multiple
- **platforms:** name(s)of the platform(s) supported. Separated by semicolons if there are multiple
- **required_age:** minimum required age according to PEGI UK standards. Many with 0 are unrated or unsupplied
- **genres:** game genres. Separated by semicolons if there are multiple
- **achievements:** number of in-game achievements, if any
- **positive_ratings:** number of positive ratings
- **negative_ratings:** number of negative ratings
- **average_playtime:** average user playtime
- **owners_lb**: the lower bound of the number of estimated owners
- **owners_ub:** the upper bound of the number of estimated owners
- **price:** current full price of the title game in GBP (pounds sterling)

After gaining a better understanding of the data, you will use Numpy, the fundamental Python library for scientific computing, and Pandas to analyze and perform operations on your dataset.

Formatting

To ensure that you are using NumPy and Pandas we require that you complete functions within a maximum number of lines. Below are some examples.

One line

```
def aFunction(parameter):
    return f"This is the {parameter}"
```

Two lines

```
def aFunction(parameter):
    var =  f"This is the {parameter}"
    return var
```

Three lines

```
def aFunction(parameter):
    var =  "This is the "
    var += f"{parameter}"
    return var
```

...

Bonus

The homework has a total of 150 points. This means you **do not** have to complete all functions in order to receive a 100. Please refer to the grading rubric for point distributions.

SECTION ONE: NumPy Arrays

Note: Before starting, refer to the grading rubric for bonus point distributions.

Function name: numpy_data_loader
Parameter(s): `file_name` (str)
Return Type: tuple of np.arrays
Description: Write a function that takes in the name of the `file_name` as a string and returns the cleaned numpy arrays corresponding to the columns.

CS2316 HOMEWORK 07: STEAM STORE ANALYZER

Columns	Datatype
appid	int
name	str
release_year	int
english	int
developer	str
platforms	str
required_age	int
genres	str
achievements	int
postive_ratings	int
negative_ratings	int
average_playtime	int
owners_lb	int
owners_ub	int
price	float

Return the numpy arrays as a tuple in the following order: appid, name, release_year, english, developer, platforms, required_age, genres, achievements, postive_ratings, negative_ratings, average_playtime, owners_lb, owners_ub, price (hint: use dtype = str)

Test Case:

```
>>> appid, name, release_year, english, developer, platforms,
required_age, genres, achievements, postive_ratings,
negative_ratings, average_playtime, owners_lb, owners_ub, price
= numpy_data_loader("steam.csv")
>>> print(developer[0])
Infinity Ward;Aspyr (Mac)
>>> print(price[:5])
array([19.99  0.   24.99 39.99  7.99])
```

Function name: age_checker

Parameter(s): `name` (np.array), `required_age` (np.array), `age` (int)

Return Type: np.array

Description: Write a function that takes in the `name` array, `required_age` array, and `age` as an int and returns an array only containing those games that are eligible to be played by the given age.

There is a **one** line maximum for this function.

CS2316 HOMEWORK 07: STEAM STORE ANALYZER

Test Case:

```
>>> name = np.array([“Wolfenstein II: The New Colossus”,
“Devil May Cry 5”, “>observer_”, “Ultimate Fishing Simulator”])
>>> required_age = np.array([18, 18, 16, 0])
>>> age_checker(name, required_age, 17)
array([“>observer_”, “Ultimate Fishing Simulator”])
```

Function name: eligible_purchases
Parameter(s): `name` (np.array), `price` (np.array), `price_conversion` (float), `budget` (int)
Return Type: np.array
Description: Write a function that takes in the `name` array, the `price` array, `price_conversion`, and `budget` and first converts the price in GBP to another currency using the conversion factor and returns an array only of the names of the games that you are eligible to purchase given your budget, assuming you only want to purchase one game.

There is a **one** line maximum for this function.
Test Case:

```
>>> name = np.array([“Microsoft Flight Simulator X: Steam
Edition”, “Ultimate Fishing Simulator”, “Far Cry - Primal”])
>>> price = np.array([19.99, 14.99, 41.99])
>>> eligible_purchases(name, price, 1.23, 50)
array([“Microsoft Flight Simulator X: Steam Edition”,
     “Ultimate Fishing Simulator”])
```

Function name: probability_of_purchasing
Parameter(s): `name` (np.array), `positive_ratings` (np.array), `negative_ratings` (np.array), `average_playtime` (np.array),
Return Type: np.array
Description: Write a function that takes in the `name` array, the `postive_ratings` array, the `negative_ratings` array, and the `average_playtime` array and returns the array of game titles that you are considering purchasing. Since you are really picky about the types of games you play, you decide to create an algorithm that calculates the probability that you will purchase a certain game. Note:
$game\ rating = \frac{positive\ rating}{positive\ rating + negative\ rating}$. Here are the criteria for your algorithm:

- Each game starts off with 0.5 probability of purchasing (hint: use np.full)
- If the corresponding game rating is:
    - >=0.9, add +0.3 to that game’s probability
    - < 0.9 and >= 0.8, add +0.2 to that game’s probability
    - Otherwise, subtract –0.25 to that game’s probability
- If the corresponding average playtime is:
    - >= 100, add +0.2 to that game’s probability
    - < 100 and >= 95, add +0.1 to that game’s probability
    - Otherwise, subtract –0.25 to that game’s probability

Return the game titles where the game has greater than or equal to 75% chance of being purchased.

Test Cases:

```
>>> name = np.array(["Manhunt", "Crystalline", "Pro Cycling
Manager 2017"])
>>> positive_ratings = np.array([816, 771, 194])
>>> negative_ratings = np.array([433, 23, 106])
>>> average_playtime = np.array([99, 982, 999])
>>> probability_of_purchasing(name, positive_ratings,
negative_ratings, average_playtime)
array(["Crystalline"])
```

SECTION TWO: Pandas Dataframes

Note: Before starting, refer to the grading rubric for bonus point distributions.

Function name: pandas_data_loader
Parameter(s): `file_name` (str)
Return Type: pd.DataFrames
Description: Write a function that takes in the name of the `file_name` as a string and returns the pandas dataframe representing the csv file. The dataframe should have **appid** as the index.
There is a **one** line maximum for this function.
Test Cases:

```
>>> pandas_data_loader("steam.csv")
      name                        release_year   ...  owners_lb    owners_ub    price
appid

1     Call of Duty®: Modern Warfare® 2  2009     ...   5000000     10000000     19.99
2     Awesomenauts - the 2D moba         2012    ...   2000000     5000000      0.00
3     Pro Cycling Manager                2017    ...     50000     100000        24.99
...
200    IOSoccer                          2018     ...     100000     200000        0.00
```

Function name: add_free
Parameter(s): `df` (pd.DataFrames)
Return Type: pd.DataFrames
Description: Write a function that takes in the pandas dataframe df and adds a column called "free" to df. This column should attain booleans with values True if the price is 0.00 and False otherwise. Secondly, convert the "english" column to boolean values, with False replacing 0 and True replacing 1. Return the mutated df.

There is a **three** line maximum for this function. Failure to complete this function in **three** lines will result in a 50% deduction, if correct.

CS2316 HOMEWORK 07: STEAM STORE ANALYZER

Test Cases:

```
>>> df = pandas_data_loader("steam.csv")
>>> add_free(df)
                                     name  release_year  english  ... price   free
appid                                                               ...
1          Call of Duty®: Modern Warfare® 2          2009     True  ... 19.99  False
2                Awesomenauts - the 2D moba          2012     True  ...  0.00   True
3                  Pro Cycling Manager 2017          2017     True  ... 24.99  False
4      IL-2 Sturmovik: Battle of Stalingrad          2014     True  ... 39.99  False
5                                    Risen          2009     True  ...  7.99  False
...
200                               IOSoccer          2018     True  ...  0.00   True
```

Function name: add_positive_percent
Parameter(s): `df` (pd.DataFrames)
Return Type: pd.DataFrames
Description: Write a function that takes in the pandas dataframe df and adds a column called "positive_percent" to df. This column should represent the percent of ratings that are positive. Round the result to 2 decimal places and return the mutated df.

There is a **two** line maximum for this function. Failure to complete this function in **two** lines will result in a 50% deduction, if correct.

Test Cases:

```
>>> df = pandas_data_loader("steam.csv")
>>> df = add_free(df)
>>> add_positive_percent(df)
                                                name  ...  positive_percent
appid                                                 ...
1                       Call of Duty®: Modern Warfare® 2  ...             91.39
2                             Awesomenauts - the 2D moba  ...             84.96
3                               Pro Cycling Manager 2017  ...             64.67
4                   IL-2 Sturmovik: Battle of Stalingrad  ...             70.64
5                                                 Risen  ...             86.80
...                                                 ...  ...               ...
196                                         Hide and Seek  ...             33.90
197                                                 Naev  ...             75.00
198    Spider-Man: Homecoming - Virtual Reality Exper...  ...             60.87
199                                       Crazy Catman  ...             56.40
200                                           IOSoccer  ...             73.75
```

Function name: add_sales_revenue
Parameter(s): `df` (pd.DataFrames)
Return Type: pd.DataFrame
Description: Write a function that takes in the pandas dataframe `df`, that first computes the median number of owners for each game, then multiplies the median number of owners by the price. Add these values to a column titled "sales_revenue" and return the mutated data frame. Round the result to 2 decimal places and return the mutated df.

There is a **two** line maximum for this function. Failure to complete this in **two** lines will result in a 50% deduction for this function.

Test Case:

```
>>> df = pandas_data_loader("steam.csv")
>>> df = add_free(df)
>>> df = add_positive_percent(df)
>>> add_sales_revenue(df)
                                                  name  ...  sales_revenue
appid                                                   ...
1                      Call of Duty®: Modern Warfare® 2  ...    149925000.0
2                            Awesomenauts - the 2D moba  ...            0.0
3                              Pro Cycling Manager 2017  ...      1874250.0
4                    IL-2 Sturmovik: Battle of Stalingrad  ...      5998500.0
5                                              Risen  ...      5992500.0
...                                                ...  ...            ...
196                                       Hide and Seek  ...            0.0
197                                               Naev  ...            0.0
198    Spider-Man: Homecoming - Virtual Reality Exper...  ...            0.0
199                                        Crazy Catman  ...         7900.0
200                                            IOSoccer  ...            0.0
```

Function name: sales_revenue_for_genre
Parameter(s): `df` (pd.DataFrames), `genre` (str)
Return Type: float
Description: Write a function that takes in a pandas dataframe `df` that has the same data returned from **add_sales_revenue** and a string `genre` that is used to find the total sales revenue for games that contain the genre rounded to 2 decimal places. You may assume that we will provide a valid genre, however the genre comparison should not be case sensitive.
There is a **two** line maximum for this function. Failure to complete this in **two** lines will result in a 50% deduction for this function.
Test Case:

```
>>> df = pandas_data_loader("steam.csv")
>>> df = add_free(df)
>>> df = add_positive_percent(df)
>>> df = add_sales_revenue(df)
>>> sales_revenue_for_genre(df, "aCtIoN")
1180381350.0
```

Function name: operating_systems
Parameter(s): `df` (pd.DataFrames)
Return Type: pd.DataFrame
Description: Write a function that takes in the pandas dataframe `df` and creates a new data frame **only containing** columns “linux”, “max”, and “windows” with values 0 or 1 indicating game compatibility. 0 corresponds to incompatible while 1 corresponds to compatible. The index of the new dataframe should be the same as `df`.

There is a **four** line maximum for this function. Failure to complete this in **four** lines will result in a 50% deduction for this function.

Test Case:

```
>>> df = pandas_data_loader("steam.csv")
>>> operating_systems(df)
```

```
       windows  mac  linux
appid
1            1    1      0
2            1    1      1
3            1    0      0
4            1    0      0
5            1    0      0
...        ...  ...    ...
196          1    0      0
197          1    0      1
198          1    0      0
199          1    0      0
200          1    0      0
```

Function name: add_num_operating_systems
Parameter(s): `df` (pd.DataFrames)
Return Type: pd.DataFrame
Description: Write a function that takes in a pandas dataframe `df` that has the same data returned from **operating_systems** and creates a column titled "num_operating_systems" that is an integer representing the number of operating systems available for each game. Secondly, add a row with appid = "total_games" that represents the number of games available for each operating system.

There is a **three** line maximum for this function. Failure to complete this in **three** lines will result in a 50% deduction for this function.

Test Case:

```
>>> df = pandas_data_loader("steam.csv")
>>> os_df = operating_systems(df)
>>> add_num_operating_systems(os_df)
```

```
             windows  mac  linux  num_operating_systems
appid
1                  1    1      0                      2
2                  1    1      1                      3
3                  1    0      0                      1
4                  1    0      0                      1
5                  1    0      0                      1
...              ...  ...    ...                    ...
197                1    0      1                      2
198                1    0      0                      1
199                1    0      0                      1
200                1    0      0                      1
total_games      200   82     56                    338
```

CS2316 HOMEWORK 07: STEAM STORE ANALYZER

Function name: unique_developers
Parameter(s): `df` (pd.DataFrames)
Return Type: pd.Series
Description: Write a function that takes in a pandas dataframe `df` that has the same data returned from **pandas_data_loader** and finds the unique *individual* developers (note how they are currently separated by semicolons). Sort the unique developers in ascending alphabetical order then create a series that has a developer id starting at 1 and incrementing until the end.

There is a **five** line maximum for this function. Failure to complete this in **five** lines will result in a 50% deduction for this function.
Test Case:

```
>>> df = pandas_data_loader("steam.csv")
>>> unique_developers(df)

1C Game Studios         1
777 Studios             2
99 Game Studio          3
Almost Human Games      4
Anatola Howard          5
                     ...
rojekti               216
Łukasz Jakowski       217
△〇□× (Miwashiba)      218
九鼎工作室              219
甲山林娛樂股份有限公司  220
Length: 220, dtype: int64
Note: The developer names will probably look different in the command line due to
encoding, this should not affect the results on GradeScope.
```

Function name: aggregated_max_price
Parameter(s): `df` (pd.DataFrames), `group` (str)
Return Type: pd.Series
Description: Write a function that takes in the pandas dataframe `df` and filters `df` to **only include english games** then calculates the aggregated maximum price based on the aggregating criterion `group`. The value of `group` can be "release_year", "developer", "required_age", and "platforms". Return a series containing the aggregated maximum price. Before returning, you should sort the series based on the aggregated maximum price in descending order.

There is a **two** line maximum for this function. Failure to complete this in **two** lines will result in a 50% deduction for this function.
Test Case:

```
>>> df = pandas_data_loader("steam.csv")
>>> aggregated_max_price(df, "release_year")
release_year
2017    49.99
```

```
2019    44.99
2016    41.99
2014    39.99
2015    29.99
2012    29.99
2018    27.79
2009    19.99
2013    16.99
2008    15.99
2007    15.99
2011     7.99
2010     6.99
2006     4.99
Name: price, dtype: float64
>>> type(aggregated_max_price(df, "release_year"))
pandas.core.series.Series
>>> aggregated_max_price(df, "developer")
developer
Ubisoft San Francisco             49.99
CAPCOM Co. Ltd.                   44.99
Ubisoft                           41.99
1C Game Studios;777 Studios       39.99
BANDAI NAMCO Studio;SHIFT;QLOC    39.99
                                  ...
Noble Empire Corp.                 0.00
Noble Master LLC                   0.00
Playsaurus                         0.00
Gameguyz                           0.00
YFC games                          0.00
Name: price, Length: 192, dtype: float64
>>> type(aggregated_max_price(df, "developer"))
pandas.core.series.Series
```

Function name: release_year_stats
Parameter(s): `df` (pd.DataFrames), `is_free` (bool)
Return Type: pd.DataFrame
Description: Write a function that takes in a pandas dataframe `df` that has the same data returned from **add_sales_revenue** and a boolean `is_free` that is used to filter the data to only include free or not free games. After filtering df to include free or not free games, create a new dataframe with index of each unique release_year and columns "avg_achievements" which represents the average achievements for each year, "avg_positive_percent" which represents the average positive percent for each year, and "num_games" which represents the total games produced that year.
There is a **four** line maximum for this function. Failure to complete this in **four** lines will result in a 50% deduction for this function.

CS2316 HOMEWORK 07: STEAM STORE ANALYZER

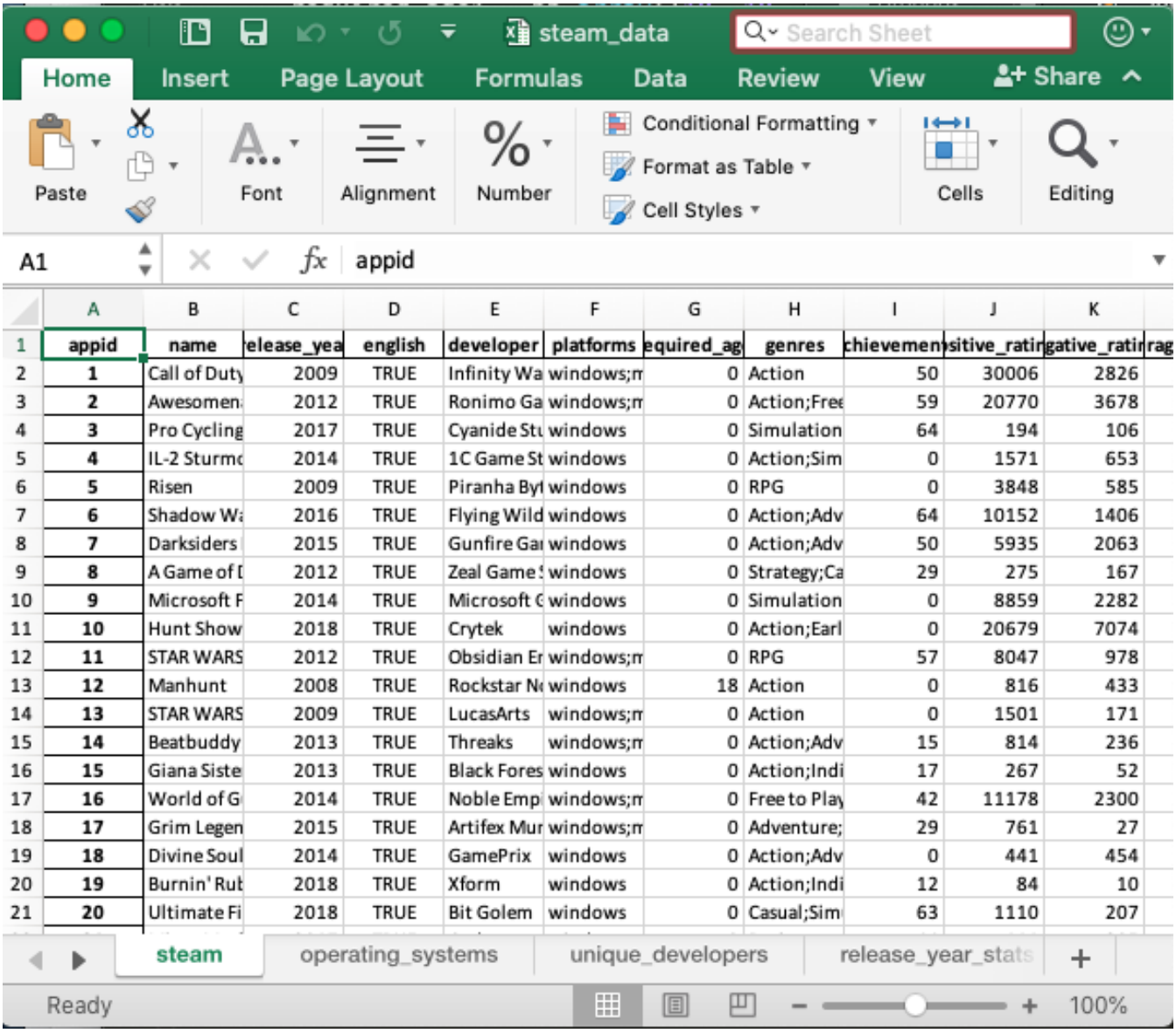
Test Case:

```
>>> df = pandas_data_loader("steam.csv")
>>> df = add_free(df)
>>> df = add_positive_percent(df)
>>> release_year_stats(df, True)
                avg_achievements  avg_positive_percent  num_games
release_year
2008                       53.00                 86.06          2
2011                       26.00                 88.12          1
2012                       59.00                 84.96          1
2013                       42.00                 73.80          1
2014                       20.83                 72.51          6
2015                       14.78                 76.17          9
2016                        3.40                 70.88         10
2017                       10.86                 63.63          7
2018                       26.75                 71.15          8
```

Function name: write_to_excel
Parameter(s): `file_name` (str), `dfs` (list), `sheet_names` (list)
Return Type: NoneType
Description: Write a function that takes in `file_name` (which includes the .xlsx extension), `dfs` a list of dataframes, and `sheet_names` a list of sheet names and writes each dataframe to the corresponding sheet_name and saves the file as `file_name`.

```
>>> df = pandas_data_loader("steam.csv")
>>> df = add_free(df)
>>> df = add_positive_percent(df)
>>> df = add_sales_revenue(df)
>>> os_df = operating_systems(df)
>>> os_df = add_num_operating_systems(os_df)
>>> ud_df = unique_developers(df)
>>> years_df = release_year_stats(df, True)
>>> write_to_excel("steam_data.xlsx", [df, os_df, ud_df,
years_df], ["steam", "operating_systems", "unique_developers",
"release_year_stats"])
```

The resulting excel file `steam_data.xlsx` should look like the following:



	A	B	C	D	E	F	G	H	I	J	K
1	appid	name	elease_yea	english	developer	platforms	equired_ag	genres	chievemen	sitive_ratir	gative_ratir	rag
2	1	Call of Duty	2009	TRUE	Infinity Wa	windows;m	0	Action	50	30006	2826	
3	2	Awesomen	2012	TRUE	Ronimo Ga	windows;m	0	Action;Free	59	20770	3678	
4	3	Pro Cycling	2017	TRUE	Cyanide Stu	windows	0	Simulation	64	194	106	
5	4	IL-2 Sturmo	2014	TRUE	1C Game St	windows	0	Action;Sim	0	1571	653	
6	5	Risen	2009	TRUE	Piranha Byt	windows	0	RPG	0	3848	585	
7	6	Shadow Wa	2016	TRUE	Flying Wild	windows	0	Action;Adv	64	10152	1406	
8	7	Darksiders	2015	TRUE	Gunfire Gar	windows	0	Action;Adv	50	5935	2063	
9	8	A Game of D	2012	TRUE	Zeal Game S	windows	0	Strategy;Ca	29	275	167	
10	9	Microsoft F	2014	TRUE	Microsoft G	windows	0	Simulation	0	8859	2282	
11	10	Hunt Show	2018	TRUE	Crytek	windows	0	Action;Earl	0	20679	7074	
12	11	STAR WARS	2012	TRUE	Obsidian En	windows;m	0	RPG	57	8047	978	
13	12	Manhunt	2008	TRUE	Rockstar No	windows	18	Action	0	816	433	
14	13	STAR WARS	2009	TRUE	LucasArts	windows;m	0	Action	0	1501	171	
15	14	Beatbuddy	2013	TRUE	Threaks	windows;m	0	Action;Adv	15	814	236	
16	15	Giana Siste	2013	TRUE	Black Fores	windows	0	Action;Indi	17	267	52	
17	16	World of G	2014	TRUE	Noble Emp	windows;m	0	Free to Play	42	11178	2300	
18	17	Grim Legen	2015	TRUE	Artifex Mur	windows;m	0	Adventure;	29	761	27	
19	18	Divine Soul	2014	TRUE	GamePrix	windows	0	Action;Adv	0	441	454	
20	19	Burnin' Rut	2018	TRUE	Xform	windows	0	Action;Indi	12	84	10	
21	20	Ultimate Fi	2018	TRUE	Bit Golem	windows	0	Casual;Sim	63	1110	207	

Bonus

- There are **50** points of total bonus for this assignment. **25** points come from the functions defined above which will be autograded through GradeScope. The other **25** points must come from 3 functions (first one is worth 5 points, last two are 10 points each) that meet the following criteria:
    - Clear and concise comment explaining what the function accomplishes and why it is practical
    - Uses pandas or numpy non-trivial functions and methods on one or more of the dataframes provided above
    - Must provide a test case and must not error

GradeScope Requirements

- **NO PRINT STATEMENTS** this will break the autograder
- **NO FUNCTION CALLS** outside of function definitions this will also break the autograder
- Make sure the file submitted is titled **HW07.py**
- **Allowed imports: numpy, pandas, pprint**
- **Only submit file HW07.py**

Grading Rubric

numpy_data_loader	10 pts
age_checker	5 pts
eligible_purchases	5 pts
probability_of_purchasing	10 pts
pandas_data_loader	5 pts
add_free	5 pts
add_postitve_percent	5 pts
add_sales_revenue	10 pts
sales_revenue_for_genre	10 pts
operating_systems	10 pts
add_num_operating_systems	10 pts
unique_developers	10 pts
aggregated_max_price	10 pts
release_year_stats	10 pts
write_to_excel	10 pts
bonus_function_1	5 pts
bonus_function_2	10 pts
bonus_function_3	10 pts
total	150/100 pts