Assignment 3: Pandas

Solve the following exercises and upload your solutions to Moodle by the due date.

Important Information!

Please try to *exactly match the outputs* provided in the examples.

Use the *exact filenames* specified for each exercise (the default suggestions from the heading). Your main code (example prints, etc.) must be guarded by `if __name__ == '__main__':`. Unless explicitly stated otherwise, you can assume correct user input and correct arguments.

You may use **only standard libraries**, plus modules covered in the lecture (including Programming in Python 1).

Exercise 1 – Submission: `a3_ex1.py` 25 Points

Write a function `analyze_fifa(data_path: str) -> pd.DataFrame` that processes the supplied file with data on a number of football players. The function should do the following:

- Read the csv file that is provided by the `data_path` argument (use `low_memory == False`).
- Replace missing values with 0.
- Compute the `position_score` for every player (see below).
- Only select the columns *short_name*, *club_position* and *position_score*, sort by *position_score* (`ascending=False`), group by *club_position* and show the top 3 players (of every category). The data frame that results from this should be returned (see below, the first 12 lines of the data frame are shown in the example output).
- The distribution of the Position Score values have to be visualized in a histogram (use 50 bins) and saved in a file (name: `position_score_distribution.pdf`) (see below).

The `position_score` has to be computed as follows:

- If the column `club_position` is `ST` (Striker):

$$position\_score = 0.4 * pace + 0.4 * shooting + 0.2 * dribbling$$

- If the column `club_position` is `CM` (Central Midfielder):

$$position\_score = 0.3 * passing + 0.3 * dribbling + 0.4 * power\_stamina$$

- If the column `club_position` is `CB` (Central Back):

$$position\_score = 0.4 * defending + 0.3 * physic + 0.3 * pace$$

- If the column `club_position` is `GK` (Goalkeeper):

$$position\_score = 0.25 * goalkeeping\_diving + 0.25 * goalkeeping\_reflexes + 0.25 * goalkeeping\_handling + 0.25 * goalkeeping\_positioning$$

- For all other positions, the `position_score` is set to `np.nan`.

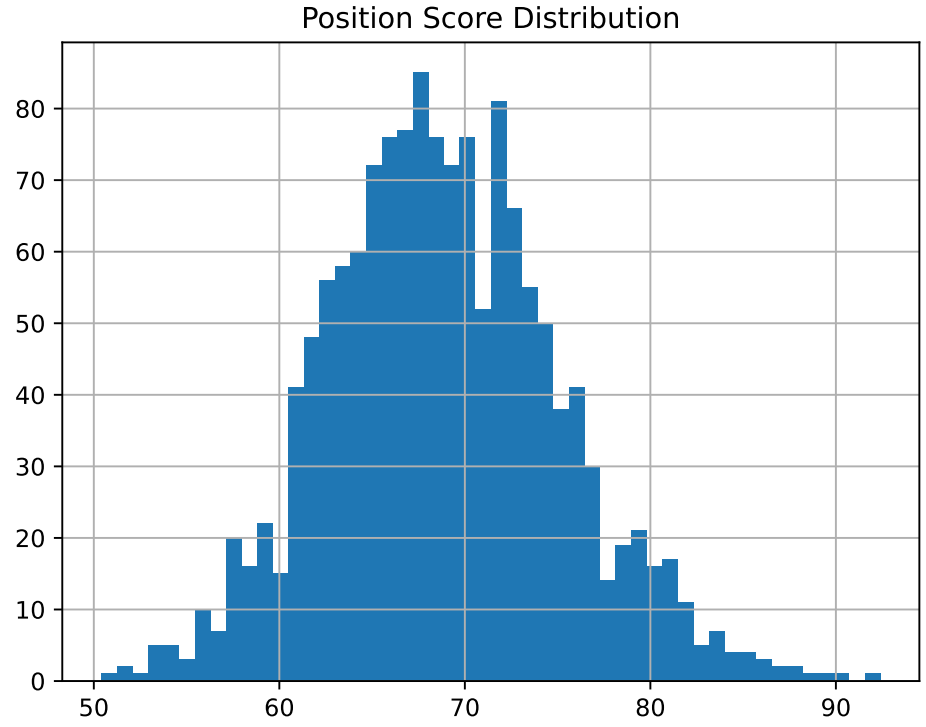
Hint: the `position_score` for every player has to be added to the data frame as a separate column.

Example code:

```
if __name__ == "__main__":
    data_path = 'players_22.csv'
    top_players = analyze_fifa(data_path)
    print(top_players[0:12]) # only prints the first 12 lines, rest should be NaN
```

Example output:

```
             short_name club_position  position_score
6             K. Mbappé            ST           92.40
2     Cristiano Ronaldo            ST           90.00
5               J. Oblak            GK           89.75
7               M. Neuer            GK           88.25
8         M. ter Stegen            GK           87.75
81        P. Aubameyang            ST           85.60
413          R. Pereyra            CM           82.80
159           S. Coates            CB           80.10
2135          M. Grimes            CM           80.10
84           S. de Vrij            CB           80.10
2772            P. Berg            CM           80.00
130          J. Giménez            CB           79.10
```



Exercise 2 – Submission: `a3_ex2.py` 25 Points

Write a function `analyze_superstore(data_path: str) -> pd.DataFrame` that analyzes data about orders and returns of our company. The function should do the following:

- Read the excel file with the provided `data_path`. One sheet is named *Orders*, a second one *Returns*. Read those into separate data frames (drop duplicate rows from returns).
- Set the column *Row ID* as the index for the data frame *Orders*.
- Merge the two data frames so that all data from *Orders* is used and the corresponding values from *Returns*. Use the indicator variable to create a new column `_merge` so that information on the source of each row is included.
- Set the value of the column *Returned* in the resulting data frame to `True`, if the indicator column has the value `both`.
- Create a view of the table that shows the total sales (i.e. sum) of the product categories depending on their return status. It should look like this:

```
Returned                False       True
Category
Furniture         682780.6204  59219.1749
Office Supplies   670470.1030  48576.9290
Technology        763445.8590  72708.1740
```

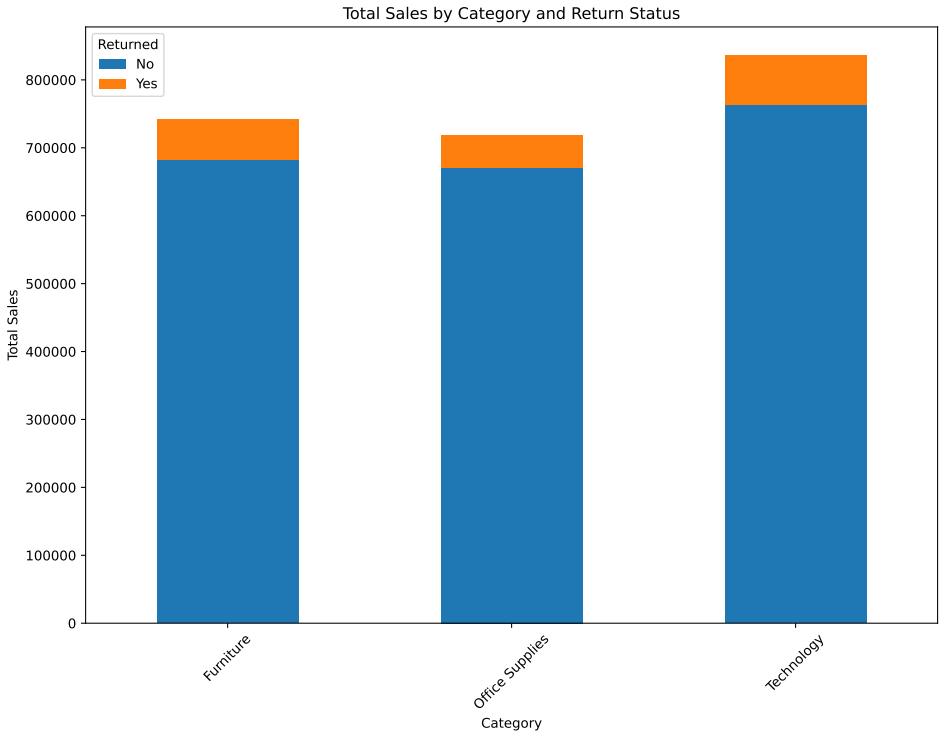
- Then create a grouping of the data where we see for every region the sum of the sales per month. Reconfigure the data frame as shown in the example output as this has to be returned by the function. Hint: to create this you will need to modify the data type of *Order Date*. Also, check out the function `dt.to_period()` that can be applied to obtain monthly intervals (which can be stored in a new column *YearMonth*)
- The two created views should also be visualized in a figure that contains two plots and saved in a file (`superstore_plots.pdf`), see below. Note: due to the locale of your system, the x-axis ticks might be shown in German, English, etc. This is ok, you do not have to force a specific locale.

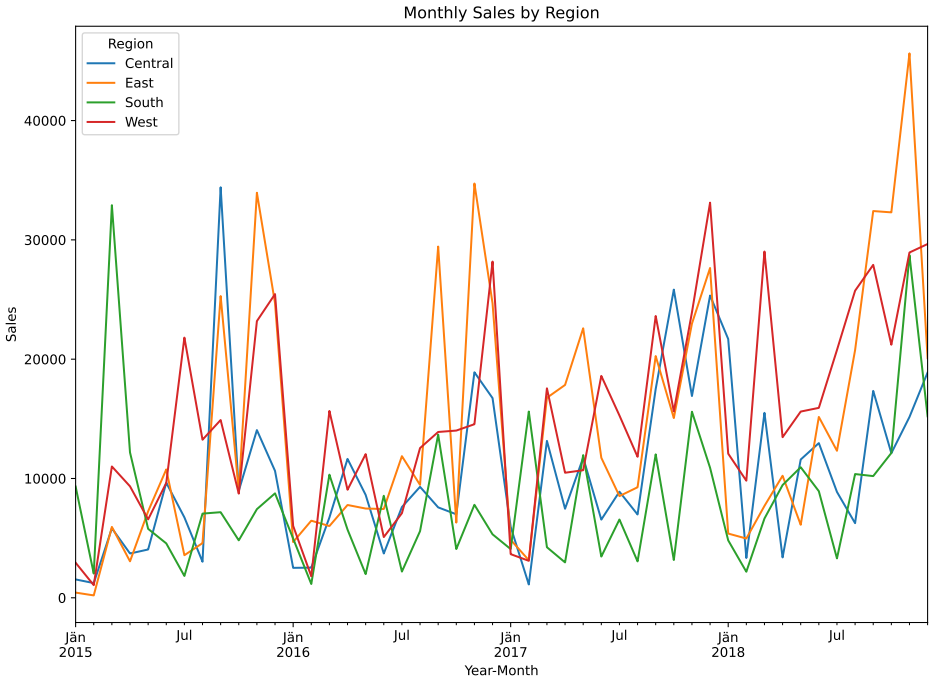
Example code:

```
if __name__ == "__main__":
    data_path = 'superstore.xlsx'
    analysis = analyze_superstore(data_path)
    print(analysis.head())
```

Example output:

```
Region      Central      East      South       West
YearMonth
2015-01    1539.906   436.174   9322.092   2938.723
2015-02    1233.174   199.776   2028.986   1057.956
2015-03    5827.602  5943.388  32911.121  11008.898
2015-04    3712.340  3054.906  12184.612   9343.487
2015-05    4048.506  7250.103   5779.240   6570.438
```


Total Sales by Category and Return Status
Returned
No
Yes
Total Sales
0
100000
200000
300000
400000
500000
600000
700000
800000
Furniture
Office Supplies
Technology
Category


Monthly Sales by Region
Region
Central
East
South
West
Sales
0
10000
20000
30000
40000
Jän
2015
Jul
Jän
2016
Jul
Jän
2017
Jul
Jän
2018
Jul
Year-Month

Exercise 3 – Submission: `a3_ex3.py` 15 Points

Write a function `analyze_titanic(data_path: str) -> pd.DataFrame:` that analyzes the number of survivors of the Titanic disaster. The function should do the following:

- Read the csv file as provided by the parameter `data_path`.
- Create a view of the data frame the shows the percentage of survivors per class and sex, like this (note that the column *Survived* is 0/1 encoded):

```
Sex       female      male
Pclass
1       0.968085  0.368852
2       0.921053  0.157407
3       0.500000  0.135447
```

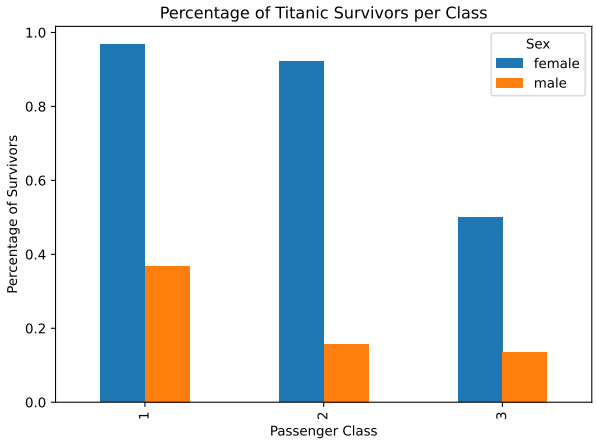
- This view should be plotted as shown below and saved in a file (`titanic_survival_rate.pdf`)
- Create another data frame that groups the total number of survivors by class and sex and reshape the output like in the example output below. This data frame must be returned by the function.

Example code:

```python
if __name__ == "__main__":
    data_path = 'titanic.csv'
    result = analyze_titanic(data_path)
    print(result)
```

Example output:

```
   Pclass     Sex  variable  value
0       1  female  Survived     91
1       1    male  Survived     45
2       2  female  Survived     70
3       2    male  Survived     17
4       3  female  Survived     72
5       3    male  Survived     47
```



Exercise 4 – Submission: `a3_ex4.py` 35 Points

Write a function `analyze_netflix(data_path: str) -> pd.DataFrame` that analyzes data about Netflix productions. The function should do the following:

- Read the csv file as specified by `data_path`.
- Adjust the data type of *date_added* to `datetime` and fill missing values in *country* with `'Unknown'`
- Compute how many new releases occurred monthly (Hint: check out `resample()`).
- Smooth the monthly releases by computing a rolling mean over a 3 month window.
- Plot the two curves (monthly releases and the smoothed trend) in a line plot as shown below and save it in a file (name: `netflix_releases_over_time.pdf`)
- Create this dictionary inside your function:

```
continent_map = {
    "United States": "North America",
    "India": "Asia",
    "United Kingdom": "Europe",
    "Japan": "Asia",
    "Brazil": "South America",
    "Unknown": "Unknown",
    "Canada": "North America",
    "France": "Europe",
    "Spain": "Europe",
    "Mexico": "North America"
}
```

- Extract from the *country* column the first country that it contains (there can be several) and store it in a new column *main_country*. Make sure to properly clean up the string.
- Map the country to a continent using the dictionary provided (fill missing values in the resulting column with `'Other'`) and store it in a new column *continent*
- Create a data frame where we group by *continent* and *main_country* and obtain the total count of releases per continent/main country. Rows should be sorted by continent ascending and within a continent descending. This should be the result:

```
        continent    main_country  count
0            Asia           India   1008
1            Asia           Japan    259
4          Europe  United Kingdom    628
2          Europe          France    212
3          Europe           Spain    181
..            ...             ...    ...
74          Other           Syria      1
83          Other    West Germany      1
84          Other        Zimbabwe      1
85  South America          Brazil     84
86        Unknown         Unknown    831

[87 rows x 3 columns]
```

- From the grouping in the previous point, only the top three countries per continent should be kept. This is the return value of the function (see example output).

Example code:

```
if __name__ == '__main__':
    data_path = 'netflix_titles.csv'
    result = analyze_netflix(data_path)
    print(result)
```

Example output:

	continent	main_country	count
0	Asia	India	1008
1	Asia	Japan	259
4	Europe	United Kingdom	628
2	Europe	France	212
3	Europe	Spain	181
7	North America	United States	3211
5	North America	Canada	271
6	North America	Mexico	134
70	Other	South Korea	211
10	Other	Australia	117
25	Other	Egypt	112
85	South America	Brazil	84
86	Unknown	Unknown	831