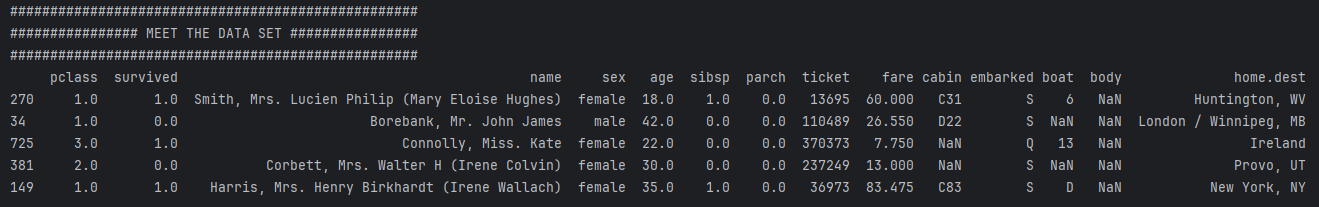
Patryk Ostrowski

Mod\_4, zad\_2 – Titanic, EDA

1. Przedstawiam losową próbkę danych:

# sniff data  
print('###################################################')  
print('################ MEET THE DATA SET ################')  
print('###################################################')  
print(df.sample(5).to\_string())  
print()



1. Zmieniam nazwy kolumn na bardziej user-friendly i ponownie przedstawiam losową próbkę danych:

# rename columns and sniff data again  
print('###################################################')  
print('################# RENAMED COLUMNS #################')  
print('###################################################')  
df.columns = ['class', 'survived', 'full\_name', 'sex', 'age', 'siblings/spouse', 'parents/children', 'ticket\_no', 'fare\_price', 'cabin\_no', 'embarked', 'boat\_no', 'body\_no', 'destination']  
print(df.sample(15).to\_string())  
print()

A screen shot of a computer screen

AI-generated content may be incorrect.

1. Przedstawiam garść faktów w kontekście zbioru danych mającego zostać poddanym analizie eksploracyjnej:

# analyze facts  
print('##################################################')  
print('##### A FEW QUICK FACTS ON THE CIRCUMSTANCES #####')  
print('##################################################')  
total\_passengers = 2200  
passengers\_in\_this\_set = len(df)  
print(f'{total\_passengers} traveled in total. This set analyses {passengers\_in\_this\_set} persons who have been found either alive or dead.')  
missing\_passengers = total\_passengers - passengers\_in\_this\_set  
print(f'What happened to {missing\_passengers} is unknown.')  
bodies\_not\_found = df['body\_no'].isnull().sum()  
survivors = (df['survived'] == 1).sum()  
print(f'{bodies\_not\_found} bodies have never been found.')  
print(f'{survivors} persons out of {passengers\_in\_this\_set} survived.')  
non\_survivors = (df['survived'] == 0).sum()  
print(f'{non\_survivors} passengers {passengers\_in\_this\_set} death has been confirmed.')  
print()

A screen shot of a computer

AI-generated content may be incorrect.

1. Przybliżam charakterystykę zestawu – jego problemy dot. wybrakowanych informacji. Jak widać takowe występują w każdej kolumnie, gdzieniegdzie w ilościach znaczących:

# data set analysis  
print('#################################################')  
print('############### DATA SET ANALYSIS ###############')  
print('#################################################')  
print()  
print('############## NULL VALUES COUNTER ##############')  
print()  
for column in df:  
 column\_sum\_of\_null = df[column].isnull().sum()  
 print(f'{column\_sum\_of\_null} times null in {column}.')

A screenshot of a computer screen

AI-generated content may be incorrect.

1. Bardzo pobieżna analiza poszczególnych kolumn – w niektórych przypadkach nieco głębsza – przeprowadzona w celu określenia co ciekawego będzie można z tych danych powyciągać.

print()  
print('############### COLUMN: CLASS ###############')  
print()  
print('Column "class" type:', df['class'].dtype)  
print('Unique values:', df['class'].unique())  
print()

A black screen with white text

AI-generated content may be incorrect.

print('############### COLUMN: SURVIVED ###############')  
print()  
print('Column "survived" type:', df['survived'].dtype)  
print('Unique values:', df['survived'].unique())

A black screen with white text

AI-generated content may be incorrect.

print('############### COLUMN: FULL NAME ###############')  
print()  
print('Column "full\_name" type:', df['full\_name'].dtype)  
print('Unique values:', df['full\_name'].unique())

A screen shot of a computer

AI-generated content may be incorrect.

print('############### COLUMN: SEX ###############')  
print()  
print('Column "sex" type:', df['sex'].dtype)  
print('Unique values:', df['sex'].unique())

A black screen with white text

AI-generated content may be incorrect.

print('############### COLUMN: AGE ###############')  
print()  
print('Column "age" type:', df['age'].dtype)  
print('Unique values:', np.sort(df['age'].unique()))

A screenshot of a computer screen

AI-generated content may be incorrect.

print('############### COLUMN: SIBLINGS/SPOUSE ###############')  
print()  
print('Column "siblings/spouse" type:', df['siblings/spouse'].dtype)  
print('Unique values:', df['siblings/spouse'].unique())

A black screen with white text

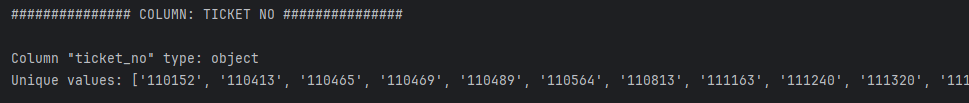
AI-generated content may be incorrect.

print('############### COLUMN: PARENTS/CHILDREN ###############')  
print()  
print('Column "parents/children" type:', df['parents/children'].dtype)  
print('Unique values:', df['parents/children'].unique())

A screen shot of a computer

AI-generated content may be incorrect.

print('############### COLUMN: TICKET NO ###############')  
print()  
print('Column "ticket\_no" type:', df['ticket\_no'].dtype)  
print('Unique values:', sorted(df['ticket\_no'].astype(str).unique()))



print('############### COLUMN: FARE PRICE ###############')  
print()  
print('Column "fare\_price" type:', df['fare\_price'].dtype)  
print('Unique values:', np.sort(df['fare\_price'].unique()))  
total\_cost\_of\_journey = df['fare\_price'].sum()  
print()  
print(f'All passengers paid {total\_cost\_of\_journey} for the journey.')  
average\_ticket\_price = df['fare\_price'].mean()  
median\_ticket\_price = df['fare\_price'].median()  
print(f'Average ticket price was {round(average\_ticket\_price, 2)} while the median was {round(median\_ticket\_price, 2)}.')  
# histogram  
df['fare\_price'].hist(bins = 200, legend=True)  
plt.title('Price per sold tickets - histogram')  
plt.xlabel('Price for ticket')  
plt.ylabel('Number of tickets sold')  
plt.show()

Na następnej stronie outcome z konsoli a jeszcze niżej histogram.

A screenshot of a computer screen

AI-generated content may be incorrect.

A graph with numbers and lines

AI-generated content may be incorrect.

Histogram pokazuje ile biletów w danym przedziale cenowym zostało sprzedanych.

print('############### COLUMN: CABIN NO. ###############')  
print()  
print('Column "cabin\_no" type:', df['cabin\_no'].dtype)  
print('Unique values:', df['cabin\_no'].unique())  
print()  
known\_cabins\_assignment\_sum = df['cabin\_no'].count()  
unknown\_cabins\_assignment\_sum = df['cabin\_no'].isnull().sum()  
print(f'{known\_cabins\_assignment\_sum} allocations to cabins have been identified. Still allocation of {unknown\_cabins\_assignment\_sum} cabins is unknown.')

A screen shot of a computer

AI-generated content may be incorrect.

print('############### COLUMN: EMBARKED ###############')  
print()  
print('Column "embarked" type:', df['embarked'].dtype)  
print('Unique values:', df['embarked'].unique())  
embarked\_from\_cherbourg = (df['embarked'] == 'C').sum()  
embarked\_from\_southampton = (df['embarked'] == 'S').sum()  
embarked\_from\_queenstown = (df['embarked'] == 'Q').sum()  
print(f'{embarked\_from\_southampton} persons onboarded in Southampton then {embarked\_from\_cherbourg} onboarded in Cherbourg and finally {embarked\_from\_queenstown} onbarded in Queenstown.')  
print()  
survivors\_cherbourg = ((df['embarked'] == 'C') & (df['survived'] == 1)).sum()  
non\_survivors\_cherbourg = ((df['embarked'] == 'C') & (df['survived'] == 0)).sum()  
print(f'From among of those who embarked in Cherbourg {survivors\_cherbourg} survived while {non\_survivors\_cherbourg} died.')  
  
survivors\_southampton = ((df['embarked'] == 'S') & (df['survived'] == 1)).sum()  
non\_survivors\_southampton = ((df['embarked'] == 'S') & (df['survived'] == 0)).sum()  
print(f'From among of those who embarked in Southampton {survivors\_southampton} survived while {non\_survivors\_southampton} died.')  
  
survivors\_queenstown = ((df['embarked'] == 'Q') & (df['survived'] == 1)).sum()  
non\_survivors\_queenstown = ((df['embarked'] == 'Q') & (df['survived'] == 0)).sum()  
print(f'From among of those who embarked in Queenstown {survivors\_queenstown} survived while {non\_survivors\_queenstown} died.')

A screenshot of a computer

AI-generated content may be incorrect.

#new data frame for chart purposes  
embarkment\_survived\_df = pd.DataFrame({  
 'embarked' : ['Cherbourg', 'Southampton', 'Queenstown'],  
 'survived' : [survivors\_cherbourg, survivors\_southampton, survivors\_queenstown],  
 'not-survived' : [non\_survivors\_cherbourg, non\_survivors\_southampton, non\_survivors\_queenstown]  
})  
  
embarkment\_survived\_df.set\_index('embarked', inplace=True)  
embarkment\_survived\_df.plot(kind='bar', stacked=True)  
plt.title('Survived/not-survived per embarkment port')  
plt.xlabel('Embarkment port')  
plt.ylabel('Number of passengers')  
plt.legend(['Survived', 'Not survived'])  
plt.tight\_layout()  
plt.show()

A graph with a bar and a number of squares

AI-generated content may be incorrect.

print('############### COLUMN: BOAT NO. ###############')  
print()  
print('Column "boat\_no" type:', df['boat\_no'].dtype)  
print('Unique values:', sorted(df['boat\_no'].astype(str).unique()))  
boats\_total = df['boat\_no'].nunique()  
print(f'There has been {boats\_total} boats in total.')  
in\_boat = df['boat\_no'].notnull().sum()  
boat\_passangers = in\_boat / boats\_total  
print(f'{in\_boat} persons got their boat which gives {int(boat\_passangers)} passengers in one boat.')  
not\_in\_boat = df['boat\_no'].isnull().sum()  
print(f'{not\_in\_boat} persons did not get their boat.')

A black background with white text

AI-generated content may be incorrect.

print('###############################################')  
print('# Check if no shitty data regarding survivors #')  
print('###############################################')  
bodies\_from\_boats = ((df['boat\_no'].notnull()) & (df['body\_no'].notnull())).sum()  
print(f'Dead from boats: {bodies\_from\_boats}.')  
bodies\_despite\_survived = ((df['survived'] == 1) & (df['body\_no'].notnull())).sum()  
print(f'Survived despite body\_no: {bodies\_despite\_survived}.')

A screenshot of a computer screen

AI-generated content may be incorrect.

print('############### COLUMN: BODY NO. ###############')  
print()  
print('Column "body\_no" type:', df['body\_no'].dtype)  
print('Unique values:', np.sort(df['body\_no'].unique()))  
total\_bodies = df['body\_no'].count()  
print(f'Bodies found in total: {total\_bodies}')

A screenshot of a computer screen

AI-generated content may be incorrect.

age\_by\_sex = df.groupby(['sex'])['age'].mean()  
print(f'Average age in each sex was:')  
print(round(age\_by\_sex, 0))  
print()  
sex\_by\_age = df.groupby(['age', 'sex']).size()  
print(f'Sum of passengers of each sex per age:')  
print(sex\_by\_age)

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

…itd.

# new data frame for scatter plot purpose  
sex\_by\_age\_df = df.groupby(['age', 'sex']).size().reset\_index(name='count')  
for gender in sex\_by\_age\_df['sex'].unique():  
 subset = sex\_by\_age\_df[sex\_by\_age\_df['sex'] == gender]  
 plt.scatter(subset['age'], subset['count'], label=gender, alpha=0.7)  
  
# scatter plot itself  
plt.xlabel('Age')  
plt.ylabel('Passengers sum')  
plt.title('Passengers sum per age by sex')  
plt.legend()  
plt.grid(True)  
plt.show()

A graph with dots and numbers

AI-generated content may be incorrect.

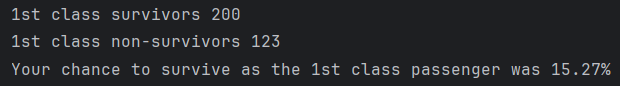
1. Analiza innych czynników zbiorczych – nie wg eksplorowanych po kolei kolumn jak powyżej. Od teraz przeprowadzona analiza skupia się na wyciągnięciu różnorakich złożonych wniosków.

# the analysis starts here  
print('###################################################')  
print('############ PURE ANALYSIS STARTS HERE ############')  
print('###################################################')  
first\_class\_passengers = (df['class'] == 1).sum()  
second\_class\_passengers = (df['class'] == 2).sum()  
third\_class\_passengers = (df['class'] == 3).sum()  
print(f'Passengers of 1st class: {first\_class\_passengers}, 2nd class: {second\_class\_passengers}, 3rd class: {third\_class\_passengers}')  
print()

A black background with white text

AI-generated content may be incorrect.

# 1st class surviving ratio analysis  
first\_class\_survivors = ( (df['class'] == 1) & (df['survived'] == 1) ).sum()  
print('1st class survivors', first\_class\_survivors)  
  
first\_class\_non\_survivors = ( (df['class'] == 1) & (df['survived'] == 0) ).sum()  
print('1st class non-survivors', first\_class\_non\_survivors)  
surviving\_ratio\_first\_class = round(first\_class\_survivors / passengers\_in\_this\_set \* 100, 2)  
print(f'Your chance to survive as the 1st class passenger was {surviving\_ratio\_first\_class}%')  
print()



# 2nd class surviving ratio analysis  
second\_class\_survivors = ( (df['class'] == 2) & (df['survived'] == 1) ).sum()  
print('2nd class survivors', second\_class\_survivors)  
  
second\_class\_non\_survivors = ( (df['class'] == 2) & (df['survived'] == 0) ).sum()  
print('2nd class non-survivors', second\_class\_non\_survivors)  
surviving\_ratio\_second\_class = round(second\_class\_survivors / passengers\_in\_this\_set \* 100, 2)  
print(f'Your chance to survive as the 2nd class passenger was {surviving\_ratio\_second\_class}%')  
print()

A black background with white text

AI-generated content may be incorrect.

# 3rd class surviving ratio analysis  
third\_class\_survivors = ( (df['class'] == 3) & (df['survived'] == 1) ).sum()  
print('3rd class survivors', third\_class\_survivors)  
  
third\_class\_non\_survivors = ( (df['class'] == 3) & (df['survived'] == 0) ).sum()  
print('3rd class non-survivors', third\_class\_non\_survivors)  
surviving\_ratio\_third\_class = round(third\_class\_survivors / passengers\_in\_this\_set \* 100, 2)  
print(f'Your chance to survive as the 3rd class passenger was {surviving\_ratio\_third\_class}%')

A black background with white text

AI-generated content may be incorrect.

# correlation  
class\_survived\_corr = df[["class", "survived"]].corr()  
print(f'Class vs. survived correlation is:')  
print(class\_survived\_corr)  
print()  
df['sex\_numeric'] = df['sex'].map({'male': 0, 'female': 1})  
sex\_survived\_corr = df[["sex\_numeric", "survived"]].corr()  
print(f'Sex vs. survived correlation is:')  
print(sex\_survived\_corr)

A screen shot of a computer

AI-generated content may be incorrect.

# new variables for new data frame for charts drawing purposes  
class\_labels = ['1st class', '2nd class', '3rd class']  
survivors\_by\_class = [first\_class\_survivors, second\_class\_survivors, third\_class\_survivors]  
non\_survivors\_by\_class = [first\_class\_non\_survivors, second\_class\_non\_survivors, third\_class\_non\_survivors]  
chance\_to\_survive = [surviving\_ratio\_first\_class, surviving\_ratio\_second\_class, surviving\_ratio\_third\_class]  
  
plot\_df = pd.DataFrame({  
 'class' : class\_labels,  
 'survived' : survivors\_by\_class,  
 'not\_survived' : non\_survivors\_by\_class,  
 'chance\_to\_survive' : chance\_to\_survive  
})  
  
plot\_df.plot(kind="pie", y="survived", labels=plot\_df["class"], legend=False)  
plt.show()  
  
plot\_df.plot(kind="bar", x="class", y=["chance\_to\_survive"])  
plt.show()

A pie chart with text on it

AI-generated content may be incorrect.

print('The cheapest tickets price (including zeros):')  
print(  
 df.groupby(['class'])['fare\_price'].min()  
)  
print()  
print('The most expensive tickets price:')  
print(  
 df.groupby(['class'])['fare\_price'].max()  
)

A screen shot of a computer

AI-generated content may be incorrect.

# new data frame to replace 0 price with None  
prices\_not\_zero\_df = df.copy()  
prices\_not\_zero\_df.loc[(prices\_not\_zero\_df['fare\_price'] == 0)] = None  
prices\_not\_zero\_df = prices\_not\_zero\_df.dropna(subset="fare\_price")  
print('New data frame with non-zeros for price:')  
print(prices\_not\_zero\_df.sample(15).to\_string())  
print()  
  
cheapest\_tickets\_by\_class = prices\_not\_zero\_df.groupby(['class'])['fare\_price'].min()  
print(  
 'The cheapest tickets by class (excluding zeros): \n', cheapest\_tickets\_by\_class  
)  
print()  
  
most\_expensive\_tickets\_by\_class = prices\_not\_zero\_df.groupby(['class'])['fare\_price'].max()  
print(  
 'The most expensive tickets by class:\n', most\_expensive\_tickets\_by\_class  
)

A screenshot of a computer program

AI-generated content may be incorrect.

# the cheapest vs. the most expensive tickets in 1st class  
print(f'1st class cheapest ticket: {cheapest\_tickets\_by\_class[1.0]}')  
print(f'1st class most expensive ticket: {most\_expensive\_tickets\_by\_class[1.0]}')  
print()  
  
# the cheapest vs. the most expensive tickets in 2n class  
print(f'2nd class cheapest ticket: {cheapest\_tickets\_by\_class[2.0]}')  
print(f'2nd class most expensive ticket: {most\_expensive\_tickets\_by\_class[2.0]}')  
print()  
  
# the cheapest vs. the most expensive tickets in 3rd class  
print(f'3rd class cheapest ticket: {cheapest\_tickets\_by\_class[3.0]}')  
print(f'3rd class most expensive ticket: {most\_expensive\_tickets\_by\_class[3.0]}')

A screen shot of a black and white screen

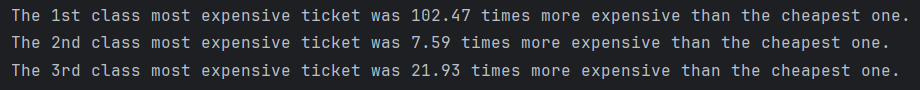
AI-generated content may be incorrect.

# how many percent was the cheapest to the most expensive in the 1st class  
cheapest\_to\_most\_expensive\_1st\_class = (cheapest\_tickets\_by\_class[1.0] / most\_expensive\_tickets\_by\_class[1.0]) \* 100  
print(f'The 1st class cheapest ticket price was {round(cheapest\_to\_most\_expensive\_1st\_class, 2)}% of the most expensive one.')  
  
# how many percent was the cheapest to the most expensive in the 2nd class  
cheapest\_to\_most\_expensive\_2nd\_class = (cheapest\_tickets\_by\_class[2.0] / most\_expensive\_tickets\_by\_class[2.0]) \* 100  
print(f'The 2nd class cheapest ticket price was {round(cheapest\_to\_most\_expensive\_2nd\_class, 2)}% of the most expensive one.')  
  
# how many percent was the cheapest to the most expensive in the 3rd class  
cheapest\_to\_most\_expensive\_3rd\_class = (cheapest\_tickets\_by\_class[3.0] / most\_expensive\_tickets\_by\_class[3.0]) \* 100  
print(f'The 3rd class cheapest ticket price was {round(cheapest\_to\_most\_expensive\_3rd\_class, 2)}% of the most expensive one.')

A screenshot of a computer screen

AI-generated content may be incorrect.

# how many times was the most expensive ticket more expensive than the cheapest in the 1st class  
most\_expensive\_to\_cheapest\_1st\_class = most\_expensive\_tickets\_by\_class[1.0] / cheapest\_tickets\_by\_class[1.0]  
print(f'The 1st class most expensive ticket was {round(most\_expensive\_to\_cheapest\_1st\_class, 2)} times more expensive than the cheapest one.')  
  
# how many times was the most expensive ticket more expensive than the cheapest in the 2nd class  
most\_expensive\_to\_cheapest\_2nd\_class = most\_expensive\_tickets\_by\_class[2.0] / cheapest\_tickets\_by\_class[2.0]  
print(f'The 2nd class most expensive ticket was {round(most\_expensive\_to\_cheapest\_2nd\_class, 2)} times more expensive than the cheapest one.')  
  
# how many times was the most expensive ticket more expensive than the cheapest in the 3rd class  
most\_expensive\_to\_cheapest\_3rd\_class = most\_expensive\_tickets\_by\_class[3.0] / cheapest\_tickets\_by\_class[3.0]  
print(f'The 3rd class most expensive ticket was {round(most\_expensive\_to\_cheapest\_3rd\_class, 2)} times more expensive than the cheapest one.')

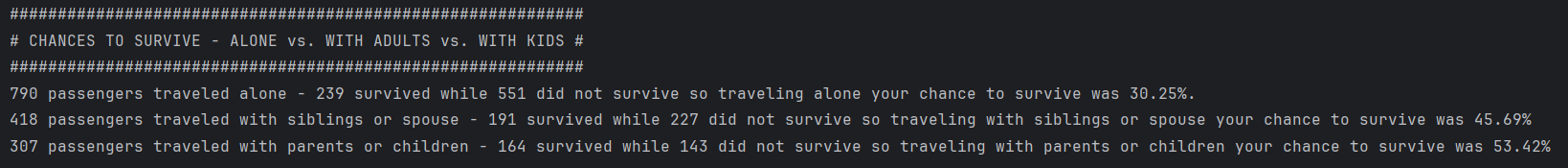


# each column from among of 'survived', 'siblings/spouse', and 'parents/children' has nulls so data is not consistent - a new data frame is needed  
print('Problematic columns:')  
print(df['survived'].isnull().sum(), 'rows is not a number for column "survived"')  
print(df['siblings/spouse'].isnull().sum(), 'rows is not a number for column "siblings/spouse"')  
print(df['parents/children'].isnull().sum(), 'rows is not a number for column "parents/children"')  
print()  
  
# new data frame created because of the above:  
not\_null\_passengers\_df = df.copy()  
not\_null\_passengers\_df = not\_null\_passengers\_df.dropna(subset=['siblings/spouse', 'parents/children'])  
  
# confirm that the data is consistent around the columns of interest  
print('Because of the above, the following data frame has been created to exclude missing data:')  
print(not\_null\_passengers\_df['survived'].isnull().sum(), 'rows is not a number for column "survived"')  
print(not\_null\_passengers\_df['siblings/spouse'].isnull().sum(), 'rows is not a number for column "siblings/spouse"')  
print(not\_null\_passengers\_df['parents/children'].isnull().sum(), 'rows is not a number for column "parents/children"')

A screen shot of a computer program

AI-generated content may be incorrect.

# conduct data mining  
print('############################################################')  
print('# CHANCES TO SURVIVE - ALONE vs. WITH ADULTS vs. WITH KIDS #')  
print('############################################################')  
traveling\_alone = ((not\_null\_passengers\_df['siblings/spouse'] == 0) & (not\_null\_passengers\_df['parents/children'] == 0)).sum()  
survivors\_traveling\_alone = ((not\_null\_passengers\_df['survived'] == 1) & (not\_null\_passengers\_df['siblings/spouse'] == 0) & (not\_null\_passengers\_df['parents/children'] == 0)).sum()  
non\_survivors\_traveling\_alone = ((not\_null\_passengers\_df['survived'] == 0) & (not\_null\_passengers\_df['siblings/spouse'] == 0) & (not\_null\_passengers\_df['parents/children'] == 0)).sum()  
chance\_to\_survive\_traveling\_alone = survivors\_traveling\_alone / traveling\_alone \* 100  
print(f'{traveling\_alone} passengers traveled alone - {survivors\_traveling\_alone} survived while {non\_survivors\_traveling\_alone} did not survive so traveling alone your chance to survive was {round(chance\_to\_survive\_traveling\_alone, 2)}%.')  
  
traveling\_with\_siblings\_spouse = (not\_null\_passengers\_df['siblings/spouse'] != 0).sum()  
survivors\_traveling\_with\_siblings\_spouse = ((not\_null\_passengers\_df['survived'] == 1) & (not\_null\_passengers\_df['siblings/spouse'] != 0)).sum()  
non\_survivors\_traveling\_with\_siblings\_spouse = ((not\_null\_passengers\_df['survived'] == 0) & (not\_null\_passengers\_df['siblings/spouse'] != 0)).sum()  
chance\_to\_survive\_traveling\_with\_siblings\_spouse = survivors\_traveling\_with\_siblings\_spouse / traveling\_with\_siblings\_spouse \* 100  
print(f'{traveling\_with\_siblings\_spouse} passengers traveled with siblings or spouse - {survivors\_traveling\_with\_siblings\_spouse} survived while {non\_survivors\_traveling\_with\_siblings\_spouse} did not survive so traveling with siblings or spouse your chance to survive was {round(chance\_to\_survive\_traveling\_with\_siblings\_spouse, 2)}%')  
  
traveling\_with\_parents\_children = (not\_null\_passengers\_df['parents/children'] != 0).sum()  
survivors\_traveling\_with\_parents\_children = ((not\_null\_passengers\_df['survived'] == 1) & (not\_null\_passengers\_df['parents/children'] != 0) ).sum()  
non\_survivors\_traveling\_with\_parents\_children = ((not\_null\_passengers\_df['survived'] == 0) & (not\_null\_passengers\_df['parents/children'] != 0)).sum()  
chance\_to\_survive\_traveling\_with\_parents\_children = survivors\_traveling\_with\_parents\_children / traveling\_with\_parents\_children \* 100  
print(f'{traveling\_with\_parents\_children} passengers traveled with parents or children - {survivors\_traveling\_with\_parents\_children} survived while {non\_survivors\_traveling\_with\_parents\_children} did not survive so traveling with parents or children your chance to survive was {round(chance\_to\_survive\_traveling\_with\_parents\_children, 2)}%')



# the youngest and the oldest survivors and non-survivors  
print('###################################################')  
print('# YOUNGEST AND OLDEST SURVIVORS AND NON-SURVIVORS #')  
print('###################################################')  
youngest\_survived = df[df['survived'] == 1]['age'].min()  
print(f'The youngest survivor was {round(youngest\_survived, 2)} years old.')  
  
oldest\_survived = df[df['survived'] == 1]['age'].max()  
print(f'The oldest survivor was {round(oldest\_survived, 2)} years old.')  
  
average\_survivor\_age = df[df['survived'] == 1]['age'].mean()  
print(f'Survivor average age was {round(average\_survivor\_age, 2)} years old.')  
  
median\_survivor\_age = df[df['survived'] == 1]['age'].median()  
print(f'Median of the survivor age was {round(median\_survivor\_age, 2)}.')  
  
print()  
  
youngest\_non\_survivor = df[df['survived'] == 0]['age'].min()  
print(f'The youngest non-survivor was {round(youngest\_non\_survivor, 2)} years old.')  
  
oldest\_non\_survivor = df[df['survived'] == 0]['age'].max()  
print(f'The oldest non-survivor was {round(oldest\_non\_survivor, 2)} years old.')  
  
average\_non\_survivor\_age = df[df['survived'] == 0]['age'].mean()  
print(f'Non-survivors average age was {round(average\_non\_survivor\_age, 2)} years old.')  
  
median\_non\_survivor\_age = df[df['survived'] == 0]['age'].median()  
print(f'Median of non-survivors age was {round(median\_non\_survivor\_age, 2)}.')

A screenshot of a computer error

AI-generated content may be incorrect.

print('##################################################')  
print('####### AGE & SUM OF SURVIVORS AT EACH AGE #######')  
print('##################################################')  
df\_clean = df[['age', 'survived']].dropna()  
df\_clean['age'] = df\_clean['age'].astype(int)  
survivors\_by\_age = df\_clean.groupby('age')['survived'].sum().astype(int)  
print(survivors\_by\_age)  
  
plt.figure(figsize=(12, 6))  
survivors\_by\_age.plot(kind='bar')  
  
plt.title('Number of survivors by age')  
plt.xlabel('Age')  
plt.ylabel('Number of survivors')  
plt.xticks(rotation=90)  
plt.grid(axis='y', linestyle='--', alpha=0.7)  
plt.tight\_layout()  
plt.show()

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a phone

AI-generated content may be incorrect.

…itd.

A graph of blue lines

AI-generated content may be incorrect.

print('##################################################')  
print('############# % OF SURVIVORS PER SEX #############')  
print('##################################################')  
women\_traveled = df[df['sex'] == 'female'].shape[0]  
print(f'{int(women\_traveled)} women traveled.')  
women\_survied = df[df['sex'] == 'female']['survived'].sum()  
print(f'{int(women\_survied)} women survived.')  
print()  
  
men\_traveled = df[df['sex'] == 'male'].shape[0]  
print(f'{int(men\_traveled)} men traveled.')  
men\_survied = df[df['sex'] == 'male']['survived'].sum()  
print(f'{int(men\_survied)} men survived.')  
print()  
  
women\_chance\_to\_survive = women\_survied / women\_traveled \* 100  
print(f'Women had {round(women\_chance\_to\_survive, 2)}% chance to survive.')  
print()  
  
men\_chance\_to\_survive = men\_survied / men\_traveled \* 100  
print(f'Men had {round(men\_chance\_to\_survive, 2)}% chance to survive.')

A screenshot of a computer screen

AI-generated content may be incorrect.

Wnioski:

1. Z dostępnych danych można wyliczyć, że obsługa statku wymagała poniesienia kosztów w wysokości co najmniej 43 550 dolarów (?) ówcześnie na sprzedaży samych tylko biletów wstępu na pokład. Nie są tu uwzględnione zyski z promocji materiałów merchandisingowych jak figurki-zabawki statków, proporczyki, naklejki itp.
2. Ilość łodzi ratunkowych była niewystarczająca.
3. Do brzegu nie dotarły osoby martwe co może sugerować, że jeśli ktokolwiek zmarł z wyziębienia pomimo uprzedniego wciągnięcia na łódź, został ostatecznie (niekoniecznie natychmiast) wrzucony z powrotem do wody.
4. Przypisania osób ocalałych do klas sprzedanych biletów a także do portów wejścia na pokład stanowią ciekawostkę analityczną – nie stanowiły raczej o fakcie przeżycia lub nieprzeżycia. Relacje osób ocalałych z innej katastrofy morskiej – zatonięcia promu Estonia – wskazują, że to gdzie akurat komu udało się znaleźć w konkretnych momentach zalania poszczególnych partii pokładu, miało nieporównanie większe znaczenie. Pasażerowie świętujący wyjątkową podróż, mogący znajdować się pod wpływem alkoholu, a dodatkowo niedowierzający temu co się dzieje, mogli mieć zupełnie inne postrzeganie powagi wydarzenia.
5. Największe znaczenie dla faktu ocalenia z katastrofy Titanica wydaje się mieć płeć – kobiety/mężczyźni zidentyfikowani wśród podróżujących: 466/843. Prawie 2x tyle mężczyzn co kobiet. Natomiast kobiety/mężczyźni zidentyfikowani wśród ocalałych: 339/161 – 72% kobiet przeżyło, podczas gdy mężczyzn przeżyło zaledwie 19%.
6. Drugim istotnym czynnikiem był fakt podróżowania z kimś – analiza pokazuje, że aż 53% szans na przeżycie miały osoby współpodróżujące z osobami w relacji rodzic-dziecko. Osoby współpodróżujące w relacji dorosły-dorosły, np. rodzeństwa lub małżeństwa, miały 45-procentową szansę na ujście z życiem. Osoby podróżujące samotnie miały 30% szansy na przeżycie. Może to wskazywać na siłę determinacji do opieki nad drugą bliską nam osobą, ale na pewno jest też wypadkową reguł pierwszeństwa dostępu do łodzi ratunkowych.

Zadanie bardzo pouczające, a wcale niewyczerpujące możliwości poznanych metod pracy na danych – zachęca do powrotu.

Całe repo dostępne tutaj: https://github.com/racibornio/Python-lessons/blob/master/akademia/zadania/mod\_4/zad\_2/mod\_4\_zad\_2.py