EDA Black Sea Initiative Vessel Movements: Vessel Inspections' Delay

It's been a year since Russia's full-scale invation of Ukraine.

The Russian invasion of Ukraine in February led to a complete halt of maritime grain shipments from Ukraine, previously a major exporter via the Black Sea. This resulted in a rise in world food prices and the threat of famine in lower-income countries. To address the issue, discussions began in April, hosted by Turkey (which controls the maritime routes from the Black Sea) and supported by the UN. The resulting agreement was signed in Istanbul on 22 July 2022, valid for a period of 120 days.

The July agreement created procedures to safely export grain from certain ports to attempt to address the 2022 food crisis. Joint Coordination and Inspection Center was set up in Turkey, with the UN serving as secretariat.
Russia suspended its participation in the agreement for several days due to a drone attack on Russian naval ships elsewhere in the Black Sea, but rejoined following mediation (for further information *see at* Wikipedia).
The Black Sea Grain Initiative was designed to free up Ukrainian wheat, barley and other food critical to nations in Africa, the Middle East and Asia, where shortages of affordable supplies sent food prices surging and helped throw more people into poverty. (see VOA)
Russia is deliberately slowing down the grain agreement by extending the length of inspections for ships heading to and from its Black Sea ports.Politico

This dataset contains the list of voyages undertaken by ships under Black Sea Graine Initiative since the 3rd of August, 2022. The dataset is downloaded from the Humanitarian Data Exchange website, open data platform managed by the United Nations Office for the Coordination of Humanitarian Affairs (OCHA) through its Centre for Humanitarian Data.

Contributor of the data Joint Coordination Centre

License Creative Commons Attribution International</i>

The data are reviewed as of 7 March 2023.

The goal of the study is to analyse the number of inspections of vessels and the waiting time of vessels (downtime).

Intro: Import and Reading Data

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from collections import Counter
```

In [2]:
```
# load dataset of outbound vessels

df_outbound = pd.read_csv("BSGI Outbound Shipments.csv")
```

Step 1: Data Understanding

- *head*
- *info*
- *describe*
- *dtypes*

In [3]:
```
df_outbound.head(2)
```

Out[3]:

	Status	#	Vessel name	IMO	Departure port	Country	Commodity	Metric tons	Departure date	I
0	Outbound	794	SANTORINI ISLAND	9393618	Chornomorsk	Spain	Wheat	44,000	7-Mar-2023	
1	Outbound	793	KARTERIA	9236092	Yuzhny/Pivdennyi	Türkiye	Soya beans	44,000	7-Mar-2023	

In [4]:
```
df_outbound.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 897 entries, 0 to 896
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Status               897 non-null    object
 1   #                    897 non-null    int64 
 2   Vessel name          897 non-null    object
 3   IMO                  897 non-null    int64 
 4   Departure port       897 non-null    object
 5   Country              897 non-null    object
 6   Commodity            897 non-null    object
 7   Metric tons          897 non-null    object
 8   Departure date       897 non-null    object
 9   Inspection cleared   866 non-null    object
 10  Income group         897 non-null    object
 11  Flag                 895 non-null    object
 12  World Bank region    897 non-null    object
 13  UN region            897 non-null    object
 14  Development category 897 non-null    object
dtypes: int64(2), object(13)
memory usage: 105.2+ KB
```

In [5]:
```
df_outbound.dtypes
```

Out[5]:
```
Status               object
#                     int64
Vessel name          object
IMO                   int64
Departure port       object
Country              object
Commodity            object
Metric tons          object
Departure date       object
Inspection cleared   object
Income group         object
Flag                 object
World Bank region    object
UN region            object
Development category object
dtype: object
```

In [6]:
```
df_outbound.isna().sum()
```

Out[6]:
```
Status               0
#                    0
Vessel name          0
IMO                  0
Departure port       0
Country              0
Commodity            0
Metric tons          0
Departure date       0
Inspection cleared   31
Income group         0
Flag                 2
World Bank region    0
UN region            0
Development category 0
dtype: int64
```

Step 2: Data Preparation

- *Renaming columns in pythonic way*
- *Managing missed values*
- *Feature creation*
- *Dropping irrelevant columns/rows*
- *Identifying duplicate rows/values*

We immediately point out the missing values in the "Flag" and "Inspection cleared" columns. In addition, there is an even significant problem exists, that is the existence of the "Outbound +" row records of the categorical variable Status along with the value "Outbound" with the same unique vessel's IMO number at the same day.
But this is a cargo that was just additionally loaded onto the ship and thus marked with an additional row (record), as a result affecting the statistics of the number of ships inspected per day. This leads to an overestimate, since the ship will be counted two or three times a day (depending on the number of additional loads).

In [7]:
```
df_outbound.columns
```

Out[7]:
```
Index(['Status', '#', 'Vessel name', 'IMO', 'Departure port', 'Country',
       'Commodity', 'Metric tons', 'Departure date', 'Inspection cleared',
       'Income group', 'Flag', 'World Bank region', 'UN region',
       'Development category'],
      dtype='object')
```

In [8]:
```
# rename column - make the columns' names pythonic

df_outbound.rename(mapper={
    'Status': 'status',
       '#': 'outbound_seq',
        'IMO': 'imo_number',
        'Vessel name': 'vessel_name',
        'Departure port': 'departure_port',
        'Country': 'destination_country',
        'Commodity': 'commodity',
        'Metric tons': 'tonnage',
        'Departure date': 'departure_date',
        'Inspection cleared': 'inspection_cleared',
        'Income group': 'income_group',
        'Flag': 'flag',
        'World Bank region': 'worldbank_region',
        'UN region': 'un_region',
        'Development category': 'dev_category'
       },
                   inplace=True, axis=1)
```

In [9]:
```
df_outbound.head(2)
```

Out[9]:

	status	outbound_seq	vessel_name	imo_number	departure_port	destination_country	commo
0	Outbound	794	SANTORINI ISLAND	9393618	Chornomorsk	Spain	W

	status	outbound_seq	vessel_name	imo_number	departure_port	destination_country	commo
1	Outbound	793	KARTERIA	9236092	Yuzhny/Pivdennyi	Türkiye	Soya b

In [10]:
```
# changing type of tonnage column from object/string to float

df_outbound.tonnage = df_outbound['tonnage'].str.replace(",", "").astype(float)

# changing dtype

df_outbound.departure_date = pd.to_datetime(df_outbound.departure_date)
df_outbound.inspection_cleared = pd.to_datetime(df_outbound.inspection_cleared)

df_outbound = df_outbound.astype({
    'outbound_seq': 'category',
    'imo_number': 'category',
    'income_group': 'category',
    'un_region': 'category',
    'income_group': 'category',
    'worldbank_region': 'category'})
```

*Rows marked **"Outbound +"** are additional shipments on the same voyage - eg two or more commodities going to the same destination, or the same commodity to two destinations. Thus we have one vessel counted twice per given day in dataset.*

*For the purposes of this analysis, we must remove all rows with status "Outbound +" by adding the concealed tonnage from these rows to the total tonnage of the vessel (**with status just 'Outbound'**).*

In [11]:
```
df_outbound[df_outbound.status == 'Outbound +'].count()
```

Out[11]:
```
status              103
outbound_seq        103
vessel_name         103
imo_number          103
departure_port      103
destination_country  103
commodity           103
tonnage             103
departure_date      103
inspection_cleared   99
income_group        103
flag                103
worldbank_region    103
un_region           103
dev_category        103
dtype: int64
```

In [12]:
```
outbound_ajusted = df_outbound.copy(deep=True)
```

In [13]:
```
vessel_imo_plus = list(outbound_ajusted[outbound_ajusted.status == 'Outbound +'].imo_number)
print("There ate {}  rows with additional shipment, i.e. shipment loaded at the same vessel ".format(len(ve
```

There ate 103 rows with additional shipment, i.e. shipment loaded at the same vessel

In [14]:
```
outbound_ajusted[outbound_ajusted.status == 'Outbound +'].head()
```

Out[14]:

	status	outbound_seq	vessel_name	imo_number	departure_port	destination_country	comm
12	Outbound +	783	MAHA ROOS	9231004	Yuzhny/Pivdennyi	China	Sunj
20	Outbound +	776	GENEVE	9442926	Yuzhny/Pivdennyi	Türkiye	I
23	Outbound +	774	ANDONIS	9763916	Yuzhny/Pivdennyi	China	Sunj
25	Outbound +	773	NEW LIBERTY	9221645	Odesa	Egypt	
32	Outbound +	767	LADY AYANA	9196395	Odesa	Egypt	Soya

In [15]:
```
outbound_ajusted[outbound_ajusted.status == 'Outbound +'].nunique()
```

Out[15]:
```
status               1
outbound_seq        92
vessel_name         87
imo_number          87
departure_port       3
destination_country 21
commodity           10
tonnage             92
departure_date      66
inspection_cleared  68
income_group         4
flag                21
worldbank_region     5
un_region            3
dev_category         2
dtype: int64
```

There are 103 record with status 'Outbound +' but only 87 unique IMO number. This mean 1) some vessels performed more than one voyage or 2) we have to deal with human mistakes. Let's inspect that

In [16]:
```
from collections import Counter

double_status = outbound_ajusted[outbound_ajusted.status == 'Outbound +'].imo_number
```

In [17]:
```
# checking vessels with additional tonnage on board by IMO number and quantity of additional shipment

additional_shipments = Counter(list(double_status)).most_common(20)
```

In [18]:
```
additional_shipments
```

Out[18]:
```
[(9400928, 3),
 (9217826, 2),
 (9629823, 2),
 (9601883, 2),
 (9221592, 2),
```

```
(9571428, 2),
(9198381, 2),
(9545510, 2),
(9703241, 2),
(9310604, 2),
(9604782, 2),
(9146962, 2),
(9171541, 2),
(9663104, 2),
(9174268, 2),
(9231004, 1),
(9442926, 1),
(9763916, 1),
(9221645, 1),
(9196395, 1)]
```

In [19]:
```
# droppin entire row in category status 'Outbound +' - subsetting additional shipment

outbound_dropped = outbound_ajusted.drop(outbound_ajusted[outbound_ajusted.status == 'Outbound +
```

In [20]:
```
# fill missed value in flag column

missed_flag = df_outbound.flag.isna()
df_outbound[missed_flag]
```

Out[20]:

	status	outbound_seq	vessel_name	imo_number	departure_port	destination_country	comm
48	Outbound	754	ROYAL STAR	9223813	Odesa	Afghanistan	V
691	Outbound	180	CENK M	7382366	Chornomorsk	Türkiye	Sunf

In [21]:
```
# fill  missed flag of the vessels with values found by unique IMO number at https://www.vesselfinder.con

df_outbound.loc[df_outbound.imo_number == 9223813, ['flag']] = "Panama"
df_outbound.loc[df_outbound.imo_number == 7382366, ['flag']] = "Comoros"
```

Subsetting records without missed inspection date *We can see that some data is missed in inspection_date column for a very logical reason. There are the vessels that are still waiting inspection at the date of analysis. This is structurally missing data (SMD). For further consideration, we will not take these rows into account.*

In [22]:
```
# subsetting vessels with cleared inspection

not_cleared = df_outbound.inspection_cleared.isna()
df_cleared = df_outbound[~not_cleared].copy()
```

In [23]:
```
num_vessels, cols = df_outbound[not_cleared].shape
print('There are {} vessels waiting inspection clearence as of March, 4 2023'.format(num_vessels))
```

```
There are 31 vessels waiting inspection clearence as of March, 4 2023
```

In [24]:
```
# subsetting 'Outbound +' rows for further calculation of additional tonnage after dropping those raws
```

```
df_outbound_plus = df_cleared[df_cleared.status == 'Outbound +'].copy()
df_outbound_plus.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 99 entries, 32 to 896
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype         
---  ------              --------------  -----         
 0   status              99 non-null     object        
 1   outbound_seq        99 non-null     category      
 2   vessel_name         99 non-null     object        
 3   imo_number          99 non-null     category      
 4   departure_port      99 non-null     object        
 5   destination_country  99 non-null     object        
 6   commodity           99 non-null     object        
 7   tonnage             99 non-null     float64       
 8   departure_date      99 non-null     datetime64[ns]
 9   inspection_cleared  99 non-null     datetime64[ns]
 10  income_group        99 non-null     category      
 11  flag                99 non-null     object        
 12  worldbank_region    99 non-null     category      
 13  un_region           99 non-null     category      
 14  dev_category        99 non-null     object        
dtypes: category(5), datetime64[ns](2), float64(1), object(7)
memory usage: 69.1+ KB
```

In [25]:

```
# subsetting 'Outbound +' rows for further calculation of additional tonnage after dropping those raws

df_outbound_plus = df_cleared[df_cleared.status == 'Outbound +'].copy()
df_outbound_plus.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 99 entries, 32 to 896
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype         
---  ------              --------------  -----         
 0   status              99 non-null     object        
 1   outbound_seq        99 non-null     category      
 2   vessel_name         99 non-null     object        
 3   imo_number          99 non-null     category      
 4   departure_port      99 non-null     object        
 5   destination_country  99 non-null     object        
 6   commodity           99 non-null     object        
 7   tonnage             99 non-null     float64       
 8   departure_date      99 non-null     datetime64[ns]
 9   inspection_cleared  99 non-null     datetime64[ns]
 10  income_group        99 non-null     category      
 11  flag                99 non-null     object        
 12  worldbank_region    99 non-null     category      
 13  un_region           99 non-null     category      
 14  dev_category        99 non-null     object        
dtypes: category(5), datetime64[ns](2), float64(1), object(7)
memory usage: 69.1+ KB
```

In [26]:

```
# Obtaining unique set for every additional shipment onboard (date, imo number, tonnage)

add_shipment = []

for index, row in df_outbound_plus.iterrows():
    date_plus = row.inspection_cleared
    imo_plus = row.imo_number
    tonnage_plus = row.tonnage
    add_shipment.append((date_plus, imo_plus, tonnage_plus))
```

```
print(add_shipment[:5])
```

```
[(Timestamp('2023-03-07 00:00:00'), 9196395, 11939.0), (Timestamp('2023-03-07 00:00:00'), 9217826, 1
1755.0), (Timestamp('2023-03-06 00:00:00'), 9584592, 20000.0), (Timestamp('2023-02-28 00:00:00'), 96
29823, 25000.0), (Timestamp('2023-03-03 00:00:00'), 9316995, 4000.0)]
```

In [27]:
```
# Check point: actual total tonnage to check after manipulation

df_cleared.tonnage.sum()
```

Out[27]:
```
22232894.0
```

In [28]:
```
df_cleared.head()
```

Out[28]:

	status	outbound_seq	vessel_name	imo_number	departure_port	destination_country	commo
31	Outbound	767	LADY AYANA	9196395	Odesa	Egypt	(
32	Outbound +	767	LADY AYANA	9196395	Odesa	Egypt	Soya be
33	Outbound	766	EAGLE TRADER	9284362	Odesa	France	Sunflo r
34	Outbound	765	AK HALIMA	9217826	Chornomorsk	Italy	Wi
35	Outbound +	765	AK HALIMA	9217826	Chornomorsk	Italy	(

In [29]:
```
# adjusting the overall tonnage of the vessel with 'Outbound +' shipment

for item in range(len(add_shipment)):

    date_plus = add_shipment[item][0]
    imo_plus = add_shipment[item][1]
    tonnage_plus = add_shipment[item][2]

    condition = (df_cleared.status == 'Outbound') \
                & (df_cleared.inspection_cleared == date_plus) \
                & (df_cleared.imo_number == imo_plus)
    df_cleared.loc[condition, ['tonnage']] = df_cleared.tonnage + tonnage_plus
```

In [30]:
```
# extra tonnage appear in total

df_cleared.tonnage.sum()
```

Out[30]:
```
23654388.0
```

In [31]:
```
# dropping "Outbound +" rows

final_df = df_cleared.drop(df_cleared[df_cleared.status == 'Outbound +'].index).reset_index(drop=True).
```

In [32]:
```
# Check point: tonnage after droping "Outbound +" with corrected overall tonnage of the voyage

final_df.tonnage.sum()
```

Out[32]: 22232894.0

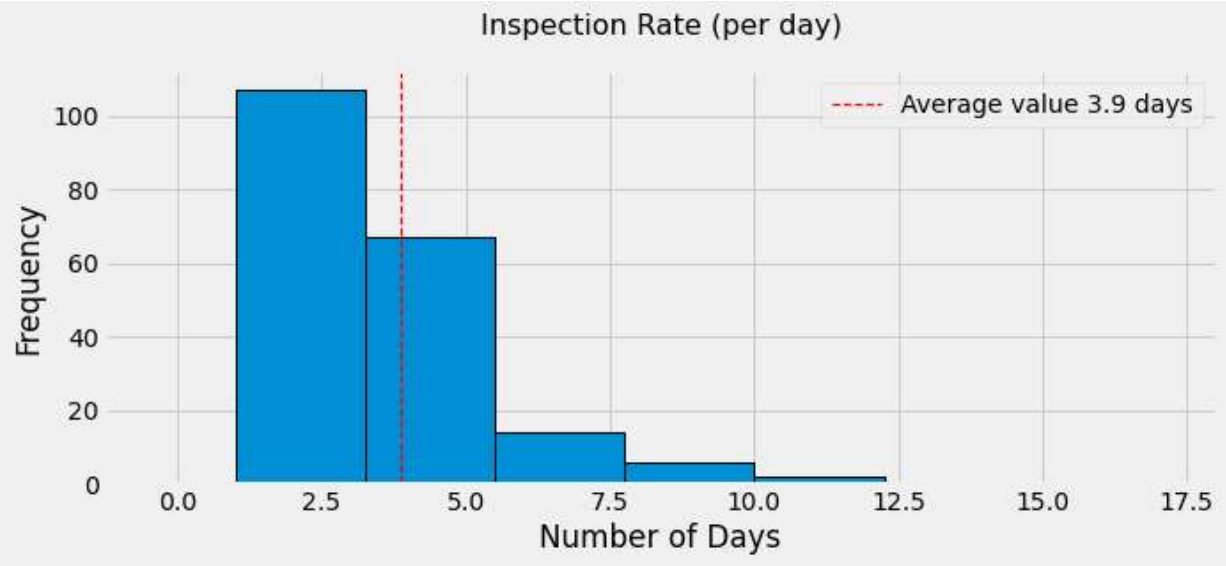
In [33]:
```
final_df['downtime'] = (final_df.inspection_cleared - final_df.departure_date).dt.days
```

Step 3: Feature Understanding

(Univariate analysis)

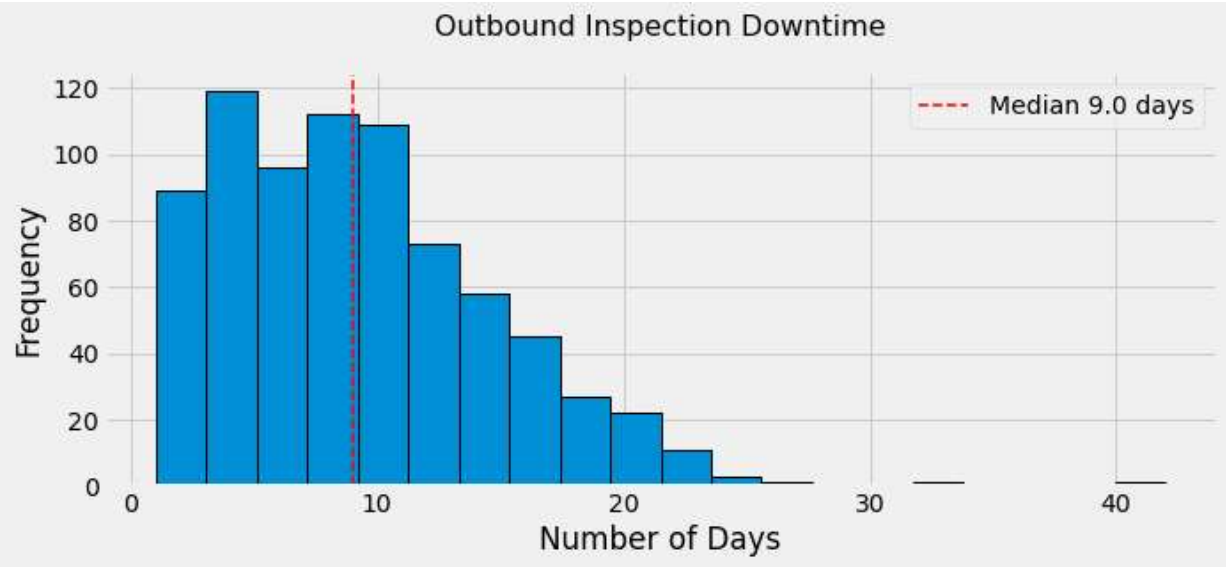
- *Ploting Feature Distribution*
  - *hystogram*
  - *boxplot*
- *Ploting time series*

In [34]:
```
# inspection rate distribution

plt.style.use('fivethirtyeight')
df_inum = final_df.inspection_cleared.value_counts().sort_index().reset_index()
df_inum.columns = ['date','number_of_inspection']

inum_avg = df_inum.number_of_inspection.mean()

ax = df_inum.number_of_inspection.hist(figsize=(10,4),
                                       bins=20,
                                       edgecolor='black',
                                       linewidth=1.)
ax.set_title('Inspection Rate (per day)',
             fontsize=16,
             pad=20)
ax.set_xlabel('Number of Days')
ax.set_ylabel('Frequency')
ax.set_xlim(right=18)
plt.axvline(inum_avg,
            color='red',
            linestyle='--',
            linewidth=1.2,
            label='Average value ' + str(round(inum_avg, 1)) + ' days')

plt.legend(fontsize=14)

plt.show()
```



In [35]:

```
# downtime distribution

downtime_median = final_df.downtime.median()
plt.style.use('fivethirtyeight')

ax = final_df.downtime.hist(figsize=(10,4),
                            bins=20,
                            edgecolor='black',
                            linewidth=1.)

ax.set_xlabel('Number of Days')
ax.set_ylabel('Frequency')
plt.axvline(downtime_median,
        linestyle='--',
        color='red',
        linewidth=1.5,
        label = 'Median ' + str(round(downtime_median,1)) + ' days')
ax.set_title('Outbound Inspection Downtime',
         fontsize=16,
         pad=20)
plt.legend(fontsize=14)
plt.show()
```



In [36]:

```
final_df[final_df.downtime < 20]['downtime'].mean()
```

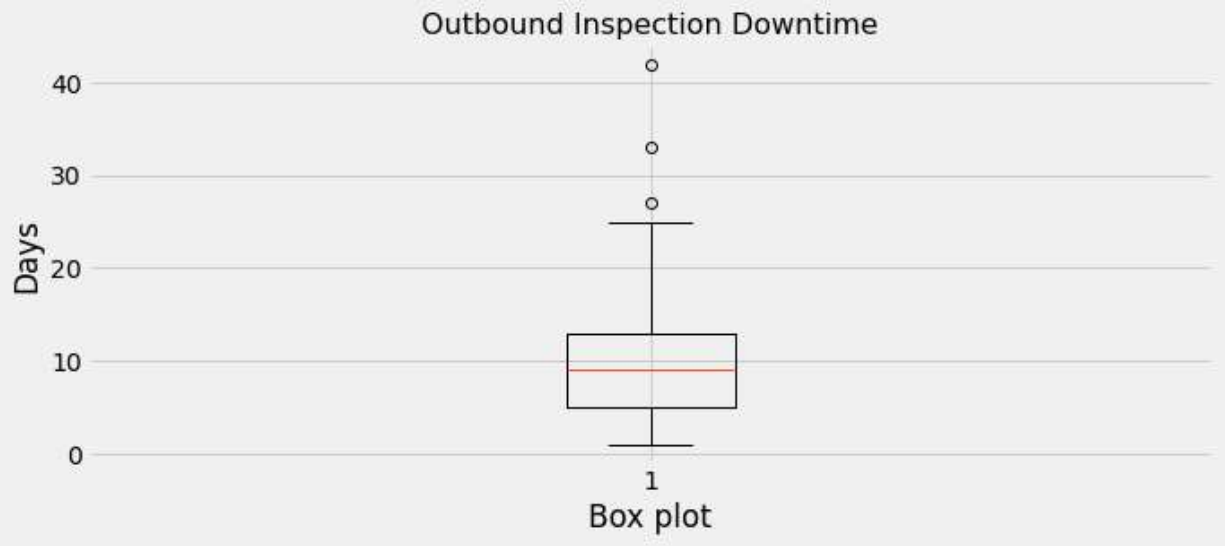
Out[36]: 8.872252747252746

In [37]:
```
inspected_mean = final_df['downtime'].mean()
```

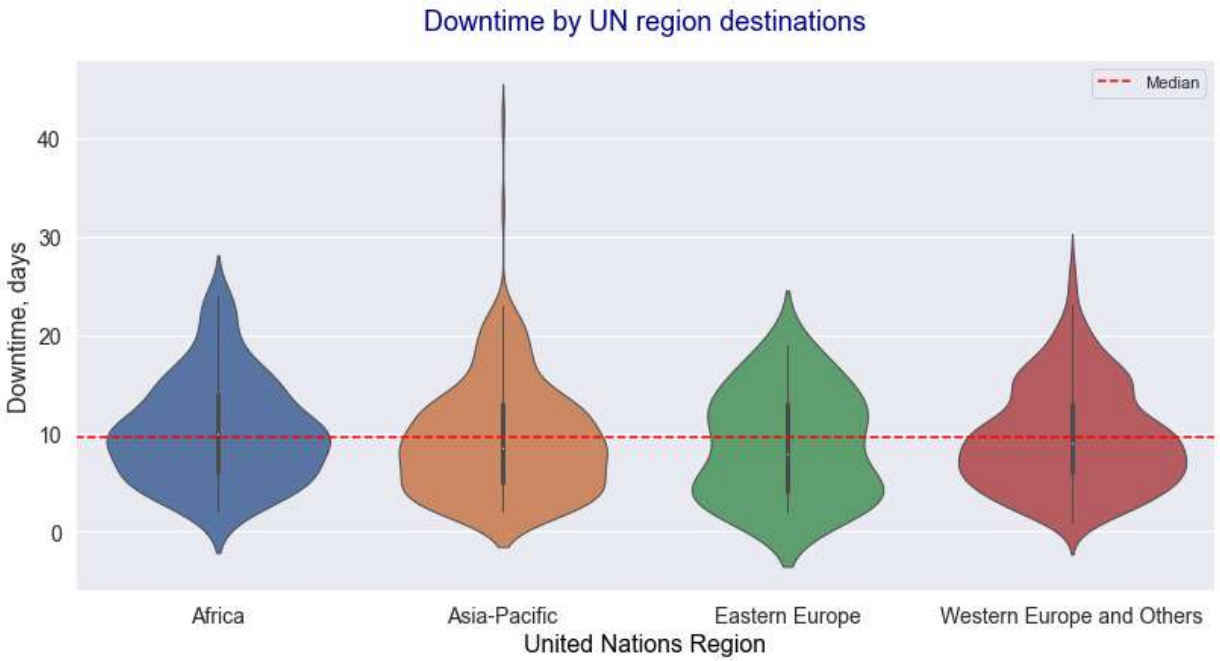
In [38]:
```
final_df.downtime.quantile(0.95)
```

Out[38]: 19.699999999999932

In [39]:
```
fig, ax = plt.subplots(figsize=(10,4))
ax = plt.boxplot(final_df.downtime)
plt.xlabel('Box plot')
plt.ylabel('Days')
plt.title('Outbound Inspection Downtime',
        fontsize=16)

plt.show()
```

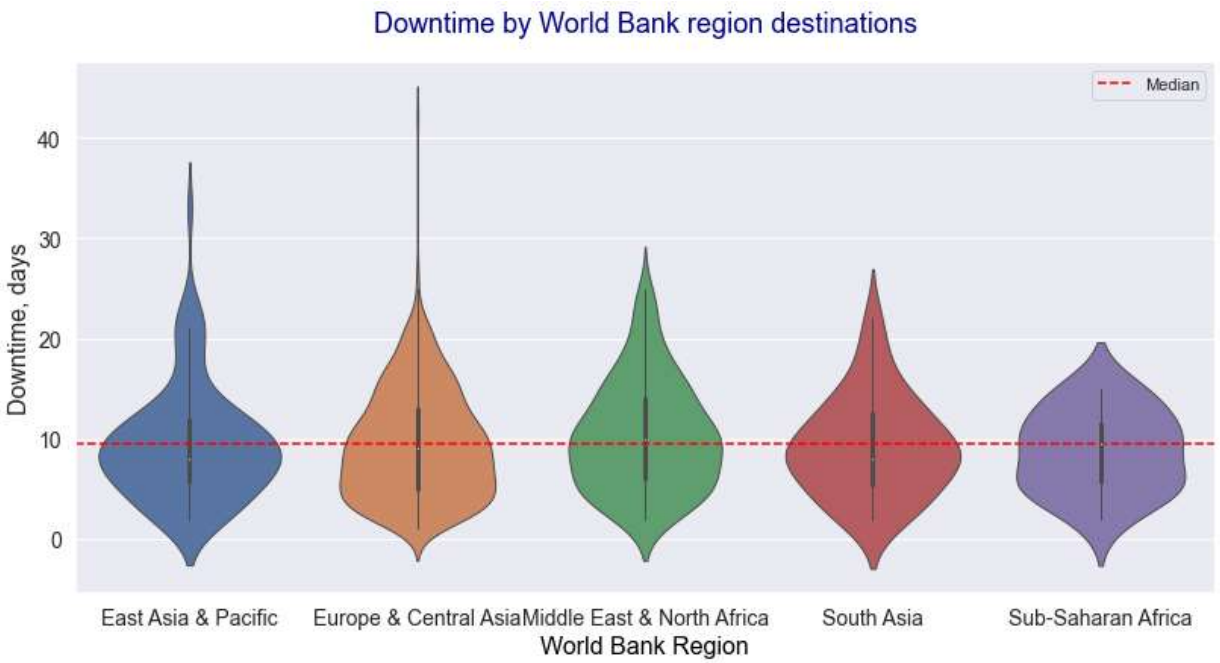


In [40]:
```
sns.set()
fig, ax = plt.subplots(figsize=(12,6))
sns.set_style("darkgrid")
sns.violinplot(ax=ax,
               data=final_df,
               x='un_region',
               y='downtime',
              linewidth = 1)
plt.title('Downtime by UN region destinations',
          color='darkblue',
          fontsize=18,
         pad=20)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
ax.axhline(inspected_mean,
           linestyle='--',
           color='red',
           linewidth=1.5,
          label = 'Median')
plt.xlabel('United Nations Region', fontsize=16, color='black')
plt.ylabel('Downtime, days', fontsize=16)
plt.legend()
plt.show()
```



In [41]:

```
fig, ax = plt.subplots(figsize=(12,6))
sns.set_style("darkgrid")
sns.violinplot(ax=ax,
               data=final_df,
               x='worldbank_region',
               y='downtime',
              linewidth = 1)
plt.title('Downtime by World Bank region destinations',
          color='darkblue',
          fontsize=18,
         pad=20)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
ax.axhline(inspected_mean,
           linestyle='--',
           color='red',
           linewidth=1.5,
          label = 'Median')
plt.xlabel('World Bank Region', fontsize=16, color='black')
plt.ylabel('Downtime, days', fontsize=16)
plt.legend()
plt.show()
```

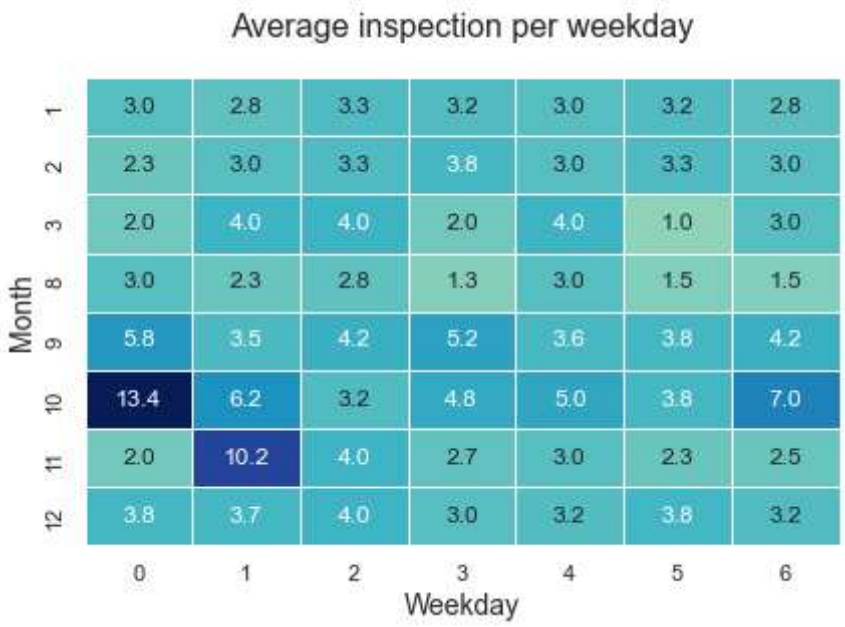


Thus, we do not see any signs of preference in delaying ship inspections depending on the region where the cargo is going.

Step 4: Feature Relationship

In [42]:
```
# Build a heatmap - average number of inspection per weekday for every months

df_reldate = final_df.inspection_cleared.value_counts().sort_index().reset_index()
df_reldate.columns = ['Date','Number of Inspection']
df_reldate['Month'] = df_reldate['Date'].dt.month
df_reldate['Weekday'] = df_reldate['Date'].dt.weekday
```

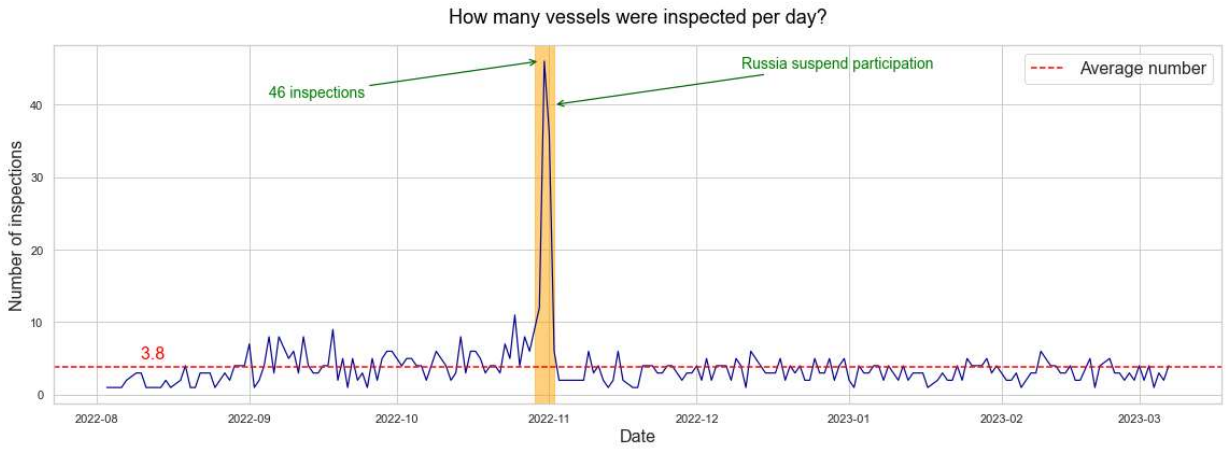
In [43]:
```
# Mean of inspections per day of week for each month

sns.heatmap(pd.crosstab(df_reldate['Month'], df_reldate['Weekday'], values=df_reldate['Number of Inspe
            annot=True,
            fmt='.1f',
            cmap='YlGnBu',
            linewidth=.5,
            cbar=False,
            center=inum_avg)
plt.xlabel('Weekday',
           fontsize=14)
plt.ylabel('Month',
           fontsize=14)
plt.title('Average inspection per weekday',
          fontsize=16,
          pad=20)
plt.show()
```



In [44]:

```
# timeseries inspections per day

df_drate = final_df.inspection_cleared.value_counts().sort_index().reset_index()
df_drate.columns = ['date','number_of_inspection']

# sns.set_style("darkgrid")
rate_avg = df_drate['number_of_inspection'].mean()
rate_max = df_drate['number_of_inspection'].max()

sns.set_style("whitegrid")
fig, ax = plt.subplots(figsize=(16, 6))
sns.lineplot(ax=ax, data=df_drate,
             x='date',
             y='number_of_inspection',
             color='darkblue',
             linewidth=1.2)

plt.title('How many vessels were inspected per day?',
          color='black',
          fontsize=18,
         pad =20)
plt.ylabel('Number of inspections', fontsize=16)
plt.xlabel('Date', fontsize=16)
plt.axhline(y=rate_avg,
            xmin=0,
            linestyle='--',
            color='red',
            label='Average number')

ax.axvspan(
    datetime(2022,10,29),
    datetime(2022,11,2),
    color='orange',
    alpha=0.5,
)

plt.annotate(
    "Russia suspend participation",
    xy=(datetime(2022,11,2), 40),
    xytext=(datetime(2022,12,10), 45),
    xycoords='data',
    color="green",
    fontsize=14,
```

```
    arrowprops=dict(arrowstyle='->',
                    color='darkgreen',
                    linewidth=1.2)
)
plt.annotate('3.8 ',
             xy=(datetime(2022,8, 10), 5),
             xytext=(datetime(2022,8, 10), 5),
              fontsize=16,
             color='red')

plt.annotate(str(rate_max) +' inspections',
              xy=(datetime(2022,10,30), 46),
             xytext=(datetime(2022,9, 5), 41),
              fontsize=14,
             color='green',
             arrowprops=dict(arrowstyle='->',
                    color='darkgreen',
                    linewidth=1.2))

plt.legend(fontsize=16)
plt.tight_layout()
plt.show()
```



Russia suspended participation in the Black Sea Initiative from 29 October to 2 November, 2022. As a result, we see a huge splash of inspections to a fantastic 46 inspections per day on 30 October 2022, against the traditional average of 3.8 inspections per day.

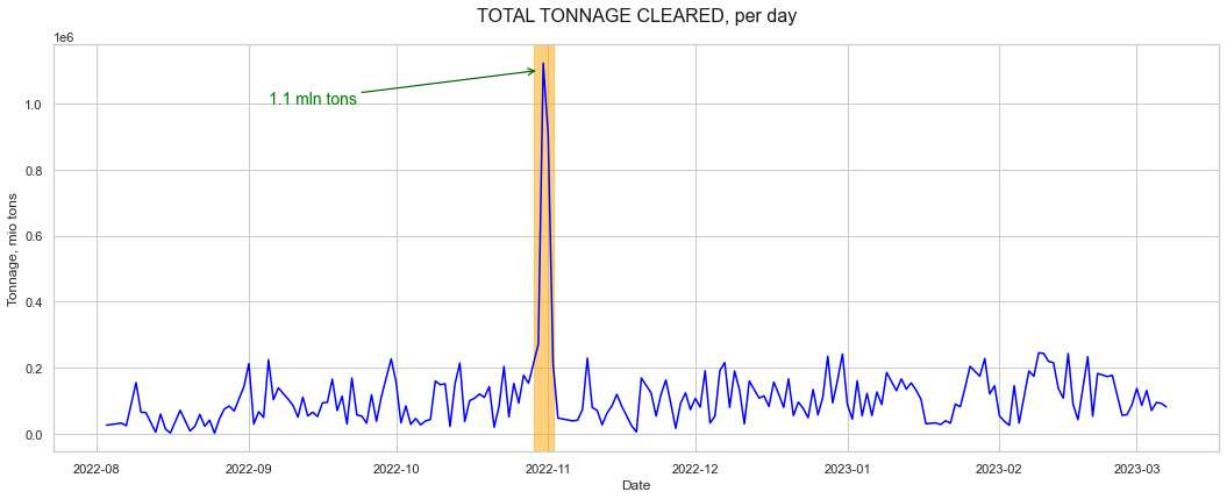
In [45]:
```
# plot total tonnage per day

df2 = final_df.groupby('inspection_cleared')['tonnage'].agg(['sum', 'count'])
tonnage_max = round(df2['sum'].max()/10**6, 1)
```

In [46]:
```
df2.head()
```

Out[46]:

inspection_cleared	sum	count
2022-08-03	26527.0	1
2022-08-06	33000.0	1
2022-08-07	25041.0	2
2022-08-09	155084.0	3
2022-08-10	65459.0	3

In [47]:
```
sns.set_style("whitegrid")
fig, ax = plt.subplots(figsize=(16, 6))
sns.lineplot(ax=ax, data=df2,
             x=df2.index,
             y='sum',
             color='blue',
             linewidth=1.5)
ax.set_title('TOTAL TONNAGE CLEARED, per day',
             fontsize=16,
             color='k',
            pad=20)
ax.set_xlabel('Date')
ax.set_ylabel('Tonnage, mio tons')

plt.annotate(str(tonnage_max) +' mln tons',
            xy=(datetime(2022,10,30), tonnage_max*10**6),
            xytext=(datetime(2022,9, 5), 1000500),
            fontsize=14,
            color='green',
            arrowprops=dict(arrowstyle='->',
                  color='darkgreen',
                  linewidth=1.2))

ax.axvspan(
    datetime(2022,10,29),
   datetime(2022,11,2),
   color='orange',
   alpha=0.5,
)
# sns.despine()
plt.show()
```



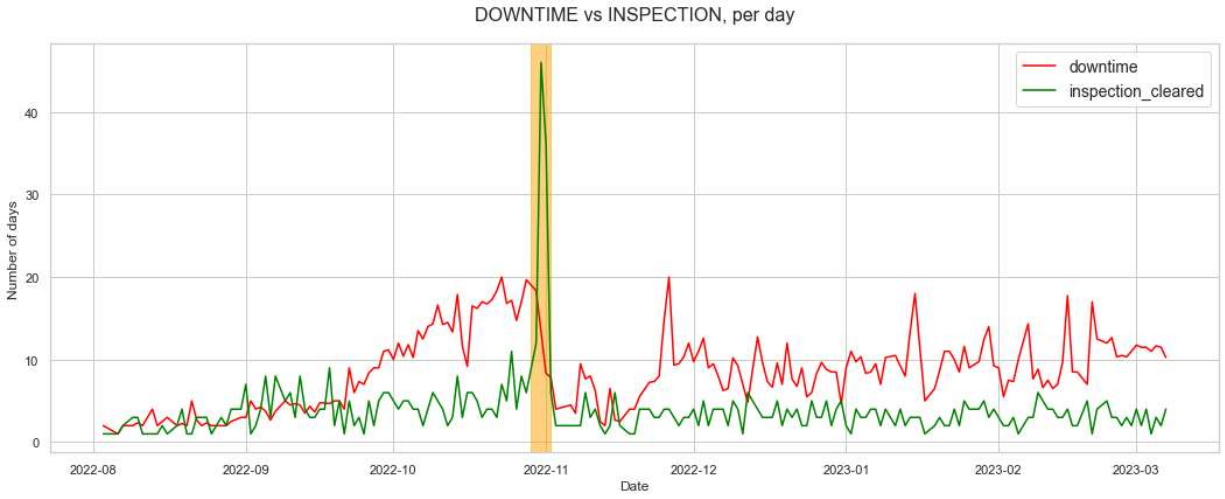
In [48]:
```
df2['sum'].mean()
```

Out[48]: 112287.34343434343

In [49]:
```
# plotting downtime mean cleared per day vs rate of inspection

df4 = final_df.groupby('inspection_cleared').agg({ 'downtime': 'mean', 'inspection_cleared': 'count'})
```

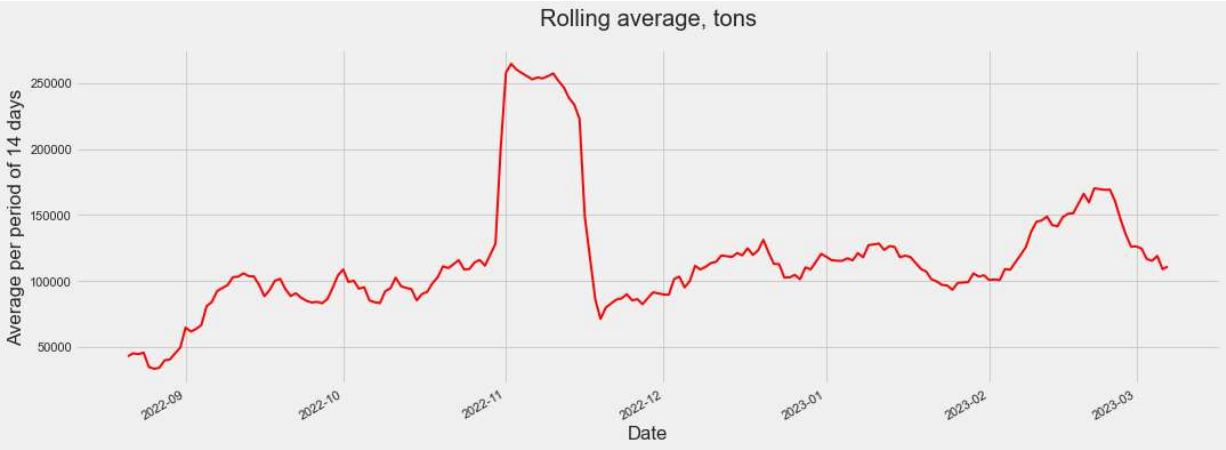
In [50]:
```
sns.set_style("whitegrid")
fig, ax = plt.subplots(figsize=(16, 6))
```

```
sns.lineplot(ax=ax, data=df4[['downtime', 'inspection_cleared']],
             linewidth=1.5,
             palette=['red', 'green'],
            dashes=False)


plt.title('DOWNTIME vs INSPECTION, per day',
          fontsize=16,
          pad=20)
plt.xlabel('Date')
plt.ylabel('Number of days')

plt.axvspan(
    datetime(2022,10,29),
    datetime(2022,11,2),
    color='orange',
    alpha=0.5,
)
plt.legend(fontsize=14)
# sns.despine()
plt.show()
```



In [62]:

```
plt.style.use('fivethirtyeight')
period = 14
df_ton_rolling = df2['sum'].rolling(window=period).mean()

df_ton_rolling.plot(figsize=(16, 6),
                    linewidth=2,
                    color='red')
plt.xlabel('Date')
plt.ylabel('Average per period of ' +str(period) + ' days')
plt.title('Rolling average, tons',
          pad=20)
plt.show()
```



In above graphs we can observe the slight tendency of shippers to increase the tonnage of ships

Step 5: Question

To what extent could the total volume of commodities deliveries increase, given the current level of inspections, if vessel tonnage increases? *We will make calculations based on the conditions of the previous period, taking into account the proposal of the Ministry of Agrarian Policy of Ukraine to increase the minimum tonnage for the transportation of oils up to 10,000 tons, for meal - 15,000 tons, for other bulk cargo - up to 25,000 tons.*

In [52]:
```
final_df.commodity.unique()
```

Out[52]:
```
array(['Corn', 'Sunflower meal', 'Wheat', 'Sunflower oil', 'Barley',
       'Soya beans', 'Rapeseed', 'Sunflower seed', 'Peas',
       'Vegetable oil', 'Canola', 'Rapeseed meal', 'Sunflower pellets',
       'Mixed', 'Wheat bran pellets', 'Sugar beet pellets'], dtype=object)
```

In [53]:
```
oil = ['Sunflover oil', 'Vegetable oil']
meal = ['Sunflower meal', 'Rapeseed meal']
```

In [54]:
```
(final_df.commodity.isin(meal)).value_counts()
```

Out[54]:
```
False    720
True      47
Name: commodity, dtype: int64
```

In [55]:
```
final_df['abstract_tonnage'] = final_df.tonnage
```

In [56]:
```
# changing tonnage of the vessels to hypotetic increased value

final_df.loc[final_df.commodity.isin(oil) & (final_df.tonnage <10000), 'abstract_tonnage'] = 10000
final_df.loc[final_df.commodity.isin(meal) & (final_df.tonnage <15000), 'abstract_tonnage'] = 15000
final_df.loc[~final_df.commodity.isin(meal) & ~final_df.commodity.isin(oil) & (final_df.tonnage <25000), 'a
```

In [57]:
```
final_df.abstract_tonnage.sum()
```

Out[57]:
```
27115783.0
```

In [58]:
```
final_df.tonnage.sum()
```

Out[58]:
```
22232894.0
```

In [59]:
```
increase_tonnage = (final_df.abstract_tonnage.sum() - final_df.tonnage.sum()) / final_df.tonnage.sum() *
```

In [60]:
```
print(f'Given the current level of inspections the vessels tonnage increase as minimum for oil up to 10 00
```

```
Given the current level of inspections the vessels tonnage increase as minimum for oil up to 10 000 ton, f
or meal - up to 15 000 tonn and for the rest of bulk cargo up to 25 000 ton, the total volume of commoditi
es deliveries under Black Sea Grain Initiative might increase by 22.0%.
```

Coda: Brief Summary

- *Average number of inspection per day is 3.8*
- *Half of the ships have a downtime of more than 9 days.*
- *There are 31 vessels waiting inspection clearance as of March, 7 2023*
- *There is no indication of preference for inspection downtime by region of freight movement.*
- *Russia suspended participation in the Black Sea Initiative from 29 October to 2 November, 2022. As a result, we see a huge splash of inspections to a fantastic 46 inspections per day on 30 October 2022, against the traditional average of 3.8 inspections per day.*
- *The number of inspections per day depends entirely on the position of Russia's representatives.*
- *Given the current level of inspections the vessels tonnage increase as minimum for oil up to 10 000 ton, for meal - up to 15 000 tonn and for the rest of bulk cargo up to 25 000 ton, the* ***total volume*** *of commodities deliveries under Black Sea Grain Initiative* ***might increase by 22.0%****.*