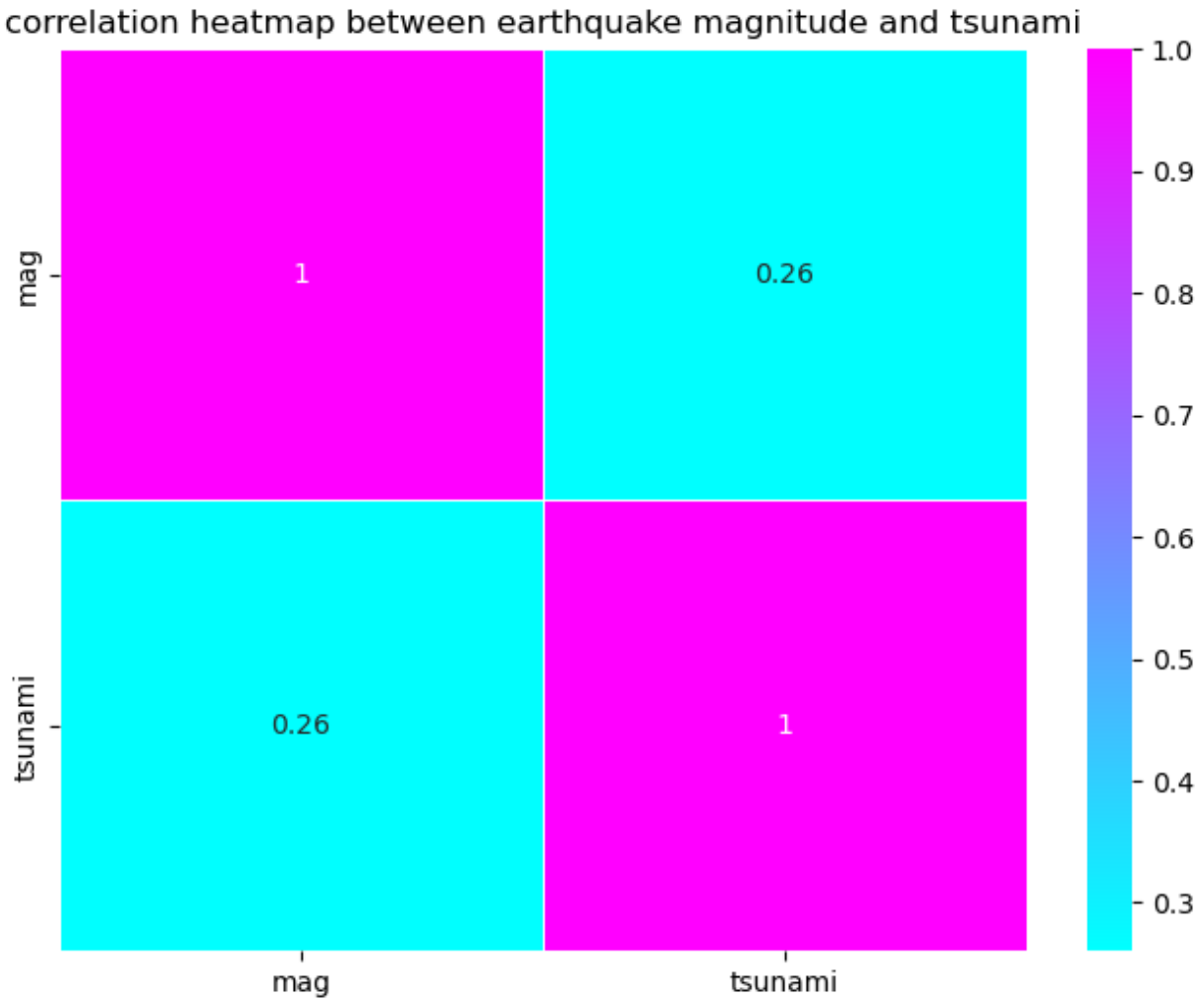
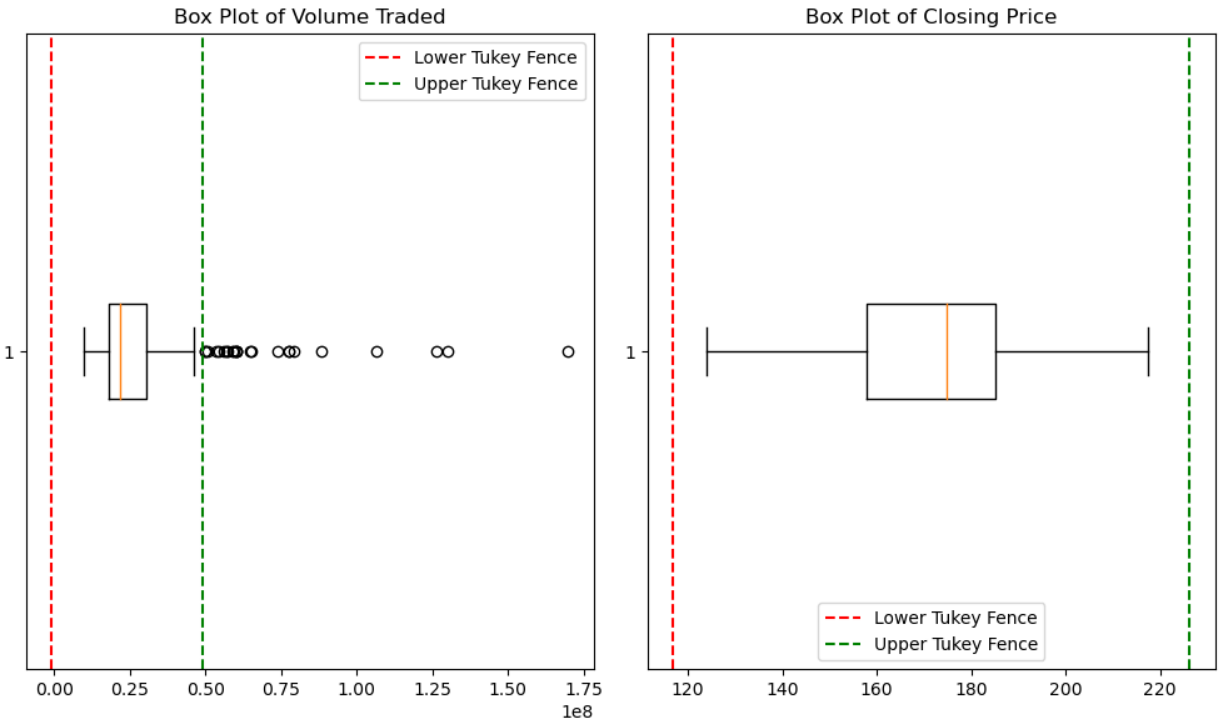
Hands-On Data Analysis with Python (2nd Edition): Page 388, Exercises 1-6,

In [8]:
```
#Using seaborn, create a heatmap to visualize the correlation coefficients betweeneart
#earthquakes (1).csv


#load sns, plt, pd
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

#earthquakes (1).csv
earthquake_data = pd.read_csv('earthquakes (1).csv')

#correlation matrix
correlation_matrix = earthquake_data[['mag', 'tsunami']].corr()

#heatmap using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='cool', linewidths=.5)

#plot
plt.title('correlation heatmap between earthquake magnitude and tsunami')
plt.show()
```



In [19]:
```
#Create a box plot of Facebook volume traded and closing prices, and draw referencelin


#load plt, pd
import matplotlib.pyplot as plt
import pandas as pd

#fb_stock_prices_2018.csv
facebook_data = pd.read_csv('fb_stock_prices_2018.csv')

#plots
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 6))

#q1 and q3 using quantile()
Q1 = facebook_data.quantile(0.25)
Q3 = facebook_data.quantile(0.75)

#IQR
IQR = Q3 - Q1

#tukey fence
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# box plots, volume Traded
axes[0].boxplot(facebook_data['volume'], vert=False)
axes[0].axvline(lower_bound['volume'], color='r', linestyle='--', label='Lower Tukey F
axes[0].axvline(upper_bound['volume'], color='g', linestyle='--', label='Upper Tukey F
axes[0].set_title('Box Plot of Volume Traded')
axes[0].legend()

#box plots closing Price
axes[1].boxplot(facebook_data['close'], vert=False)
axes[1].axvline(lower_bound['close'], color='r', linestyle='--', label='Lower Tukey Fe
axes[1].axvline(upper_bound['close'], color='g', linestyle='--', label='Upper Tukey Fe
axes[1].set_title('Box Plot of Closing Price')
axes[1].legend()

plt.tight_layout()
plt.show()
```

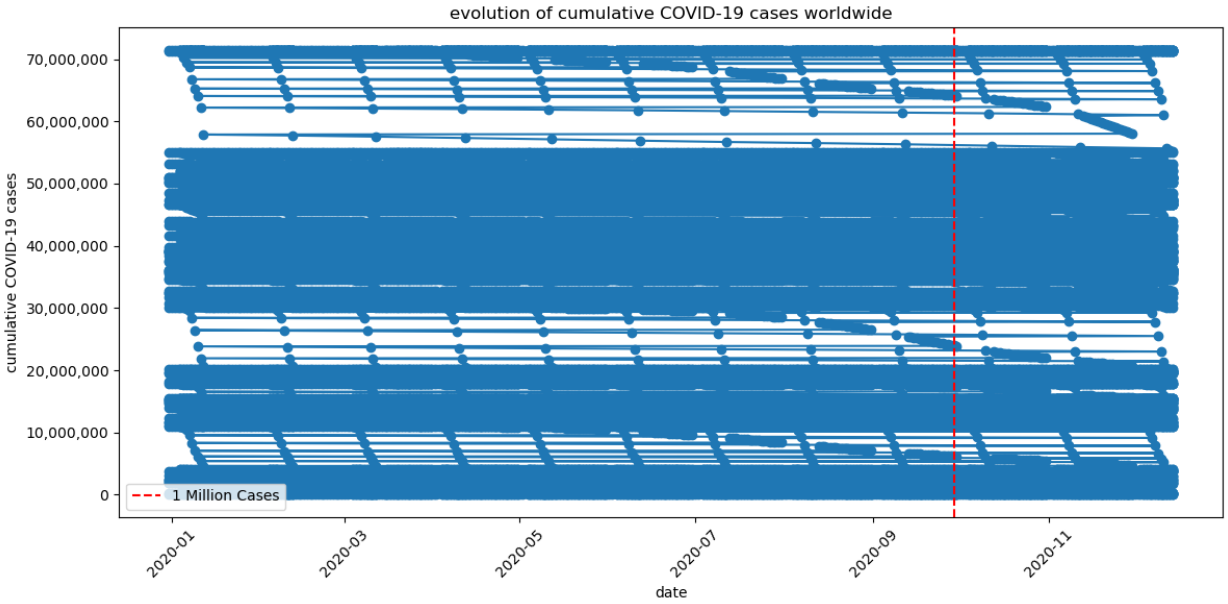
```
C:\Users\lexiw\AppData\Local\Temp\ipykernel_6280\3833168951.py:15: FutureWarning: The
default value of numeric_only in DataFrame.quantile is deprecated. In a future versio
n, it will default to False. Select only valid columns or specify the value of numeri
c_only to silence this warning.
  Q1 = facebook_data.quantile(0.25)
C:\Users\lexiw\AppData\Local\Temp\ipykernel_6280\3833168951.py:16: FutureWarning: The
default value of numeric_only in DataFrame.quantile is deprecated. In a future versio
n, it will default to False. Select only valid columns or specify the value of numeri
c_only to silence this warning.
  Q3 = facebook_data.quantile(0.75)
```



In [12]:
```
#Plot the evolution of cumulative COVID-19 cases worldwide, and add a dashedvertical l



#load plt , pd
import matplotlib.pyplot as plt
import pandas as pd


#covid 19.csv
covid_data = pd.read_csv('covid 19.csv')

#date column to datetime
covid_data['dateRep'] = pd.to_datetime(covid_data['dateRep'])

#Cumulative_number_for_14_days_of_COVID-19_cases_per_100000
covid_data['Cumulative_number_for_14_days_of_COVID-19_cases_per_100000'] = covid_data[

#date when it surpassed 1 million cases
date_1million = covid_data[covid_data['Cumulative_number_for_14_days_of_COVID-19_cases

#plot
plt.figure(figsize=(12, 6))
plt.plot(covid_data['dateRep'], covid_data['Cumulative_number_for_14_days_of_COVID-19_

#line when surpassed 1 million cases
plt.axvline(x=date_1million, color='r', linestyle='--', label='1 Million Cases')

#y-axis tick labels with commas for thousands separator
plt.gca().get_yaxis().set_major_formatter(plt.FuncFormatter(lambda x, loc: "{:,}".form

#plot
plt.xlabel('date')
plt.ylabel('cumulative COVID-19 cases')
plt.title('evolution of cumulative COVID-19 cases worldwide')
plt.legend()
```

```
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

#doesnt work, graph is wrong
```

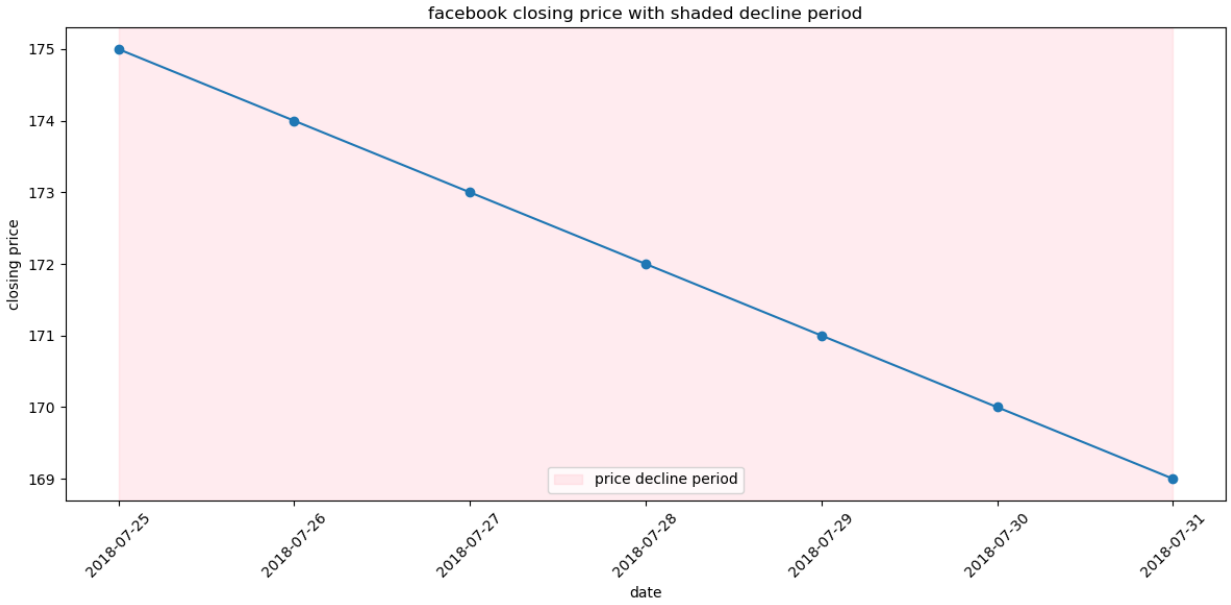
```
C:\Users\lexiw\AppData\Local\Temp\ipykernel_6280\2187644322.py:14: UserWarning: Parsi
ng dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This m
ay lead to inconsistently parsed dates! Specify a format to ensure consistent parsin
g.
  covid_data['dateRep'] = pd.to_datetime(covid_data['dateRep'])
```



In [17]:

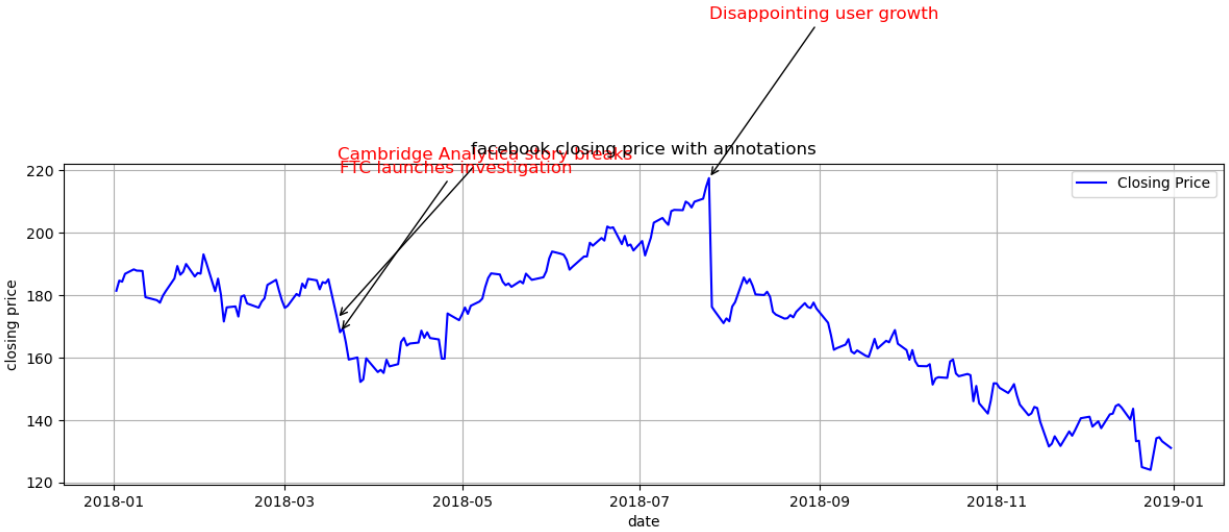
```
#Use axvspan() to shade a rectangle from '2018-07-25' to '2018-07-31',which marks the


#load plt, pd
import matplotlib.pyplot as plt
import pandas as pd

#date from 2018-07-31 to 2018-07-31
data = {'date': ['2018-07-25', '2018-07-26', '2018-07-27', '2018-07-28', '2018-07-29',
    'closing price': [175, 174, 173, 172, 171, 170, 169]}

#df
df = pd.DataFrame(data)

#date column to datetime
df['date'] = pd.to_datetime(df['date'])

#plot
plt.figure(figsize=(12, 6))
plt.plot(df['date'], df['closing price'], marker='o', linestyle='-')

#shade from 2018-07-25 to 2018-07-31
plt.axvspan('2018-07-25', '2018-07-31', alpha=0.3, color='pink', label='price decline

#plot
plt.xlabel('date')
plt.ylabel('closing price')
plt.title('facebook closing price with shaded decline period')
plt.legend()
```

```
#x axis
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



In [21]:
```
#Using the Facebook stock price data, annotate the following three events on a lineplo


import pandas as pd
import matplotlib.pyplot as plt


#fb_stock_prices_2018.csv
data = pd.read_csv('fb_stock_prices_2018.csv')

#date column to datetime
data['date'] = pd.to_datetime(data['date'])

#plot for closing price
plt.figure(figsize=(12, 6))
plt.plot(data['date'], data['close'], label='Closing Price', color='blue')

#ABC for #5
events = [('Disappointing user growth', '2018-07-25'), ('Cambridge Analytica story bre
    ('FTC launches investigation', '2018-03-20')]

for event in events:
    event_name, event_date = event
    event_date = pd.to_datetime(event_date)
    event_price = data[data['date'] == event_date]['close'].values[0]

    plt.annotate(event_name, xy=(event_date, event_price), xytext=(event_date, event_p
                 arrowprops=dict(arrowstyle='->'), fontsize=12, color='red', ha='left'

#plot
plt.xlabel('date')
plt.ylabel('closing price')
plt.title('facebook closing price with annotations')
```

```
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [31]:
```
#Modify the reg_resid_plots() function to use a matplotlib colormapinstead of cycling


#load sns, plt and np
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

def reg_resid_plots(x, y, data, color=None, label=None):
    #regression model
    reg = sns.regplot(x=x, y=y, data=data, color=color, label=label, scatter_kws={"s":
    x_pred = reg.get_lines()[0].get_xdata()
    y_pred = reg.get_lines()[0].get_ydata()
    residuals = y - np.interp(x, x_pred, y_pred)

    #qualitative colormap
    cmap = plt.get_cmap('Set1')

    #scatter plot
    plt.scatter(x, residuals, c=residuals, cmap=cmap, s=10)
    plt.xlabel(x)
    plt.ylabel('Residuals')
    plt.title(f'Residual Plot for {x} vs. {y}')

    #colorbar?
    cbar = plt.colorbar()
    cbar.set_label('residual value')

    #plot
    plt.legend()
    plt.show()
```

In [ ]:
```

```