МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Практикум №7

з курсу «Аналіз даних в інформаційнних системах» на тему: « Аналіз часових послідовностей»

Викладач:

Ліхоузова Т.A.

Виконав:

студент 2 курсу групи ІП-15 ФІОТ  
Мєшков Андрій Ігорович

Київ-2023

**Практикум №7**

**Аналіз часових послідовностей**

**Мета роботи**: ознайомитись з методами моделювання часових послідовностей.

**Завдання:**

Скачати потрібні дані.

**Основне завдання**

1. Побудувати та проаналізувати часовий ряд для статистики захворювань на Covid в двох сусідніх країнах по вашому вибору (дані взяти в інтернеті).
2. Побудувати та проаналізувати часовий ряд для курсу гривня/долар або гривня/євро за останні 3 роки (дані взяти в інтернеті).

**Додаткове завдання**

Потрібно з'ясувати, чи є сезонна компонента в кількості опадів в Сіетлі. (https://www.kaggle.com/rtatman/did-it-rain-in-seattle-19482017/data,   або seattleWeather\_1948-2017.csv [Скачати потрібні дані](https://drive.google.com/drive/folders/1kYY4pxkeLXoszCX8icpFUrhFCnwgAyhz?usp=sharing)).

1. Градуси перевести в Цельсії.
2. Чи є кореляція між температурою та опадами?
3. Скласти прогноз опадів на 2018 рік, оцінити точність прогнозу

**Хід роботи:**

**Основне завдання:**

*Імпортуємо потрібні бібліотеки.*

*# Fetch the Covid-19 data for Germany and Poland from the "Our World in Data" website*

*import* pandas *as* pd

*import* seaborn *as* sns

*import* statsmodels.api *as* sm

*import* statsmodels.tsa.api *as* smt

*from* statsmodels.graphics.tsaplots *import* plot\_acf, plot\_pacf

*Зчитаємо файл.*

covid\_data = pd.read\_csv('data.csv', sep=',', decimal=',', encoding='windows-1251')

poland\_covid = covid\_data[covid\_data['countriesAndTerritories'] == 'Poland']

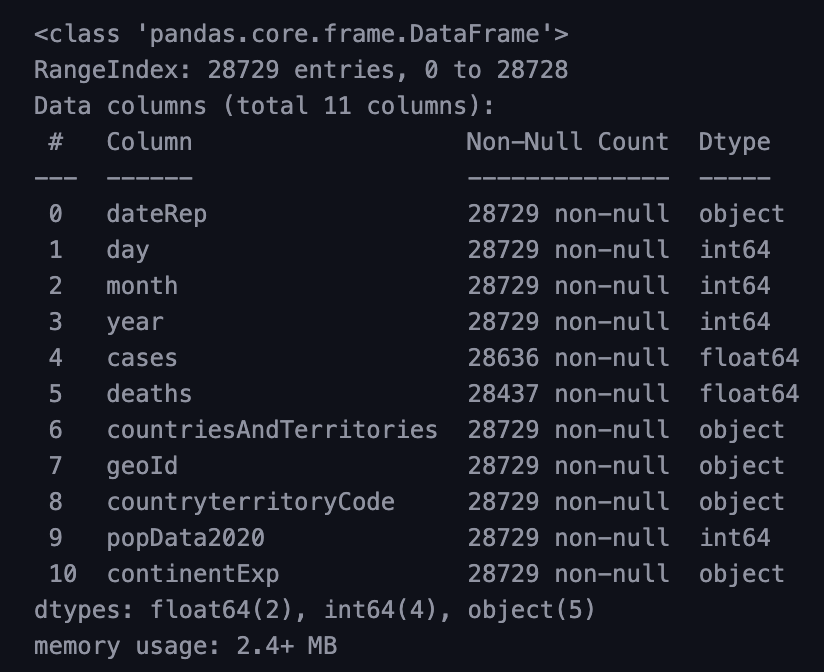
germany\_covid = covid\_data[covid\_data['countriesAndTerritories'] == 'Germany']

poland\_covid['dateRep'] = pd.to\_datetime(p\_covid['dateRep'], format='%d/%m/%Y')

germany\_covid['dateRep'] = pd.to\_datetime(g\_covid['dateRep'], format='%d/%m/%Y')

*Проаналізуємо структуру.*

covid\_data.info()



*Побудуймо щоденні нові підтверджені випадки для Польщі та Німеччини.*

*import* matplotlib.pyplot *as* plt

*import* matplotlib.dates *as* mdates

*# Plot the daily new confirmed cases for Poland and Germany*

fig, ax = plt.subplots(figsize=(15, 6))

ax.plot(poland\_covid['dateRep'], poland\_covid['cases'], label='Poland')

ax.plot(germany\_covid['dateRep'], germany\_covid['cases'], label='Germany')

ax.set\_title('Daily new confirmed cases of Covid-19')

locator = mdates.AutoDateLocator()

formatter = mdates.AutoDateFormatter(locator)

ax.xaxis.set\_major\_locator(locator)

ax.xaxis.set\_major\_formatter(formatter)

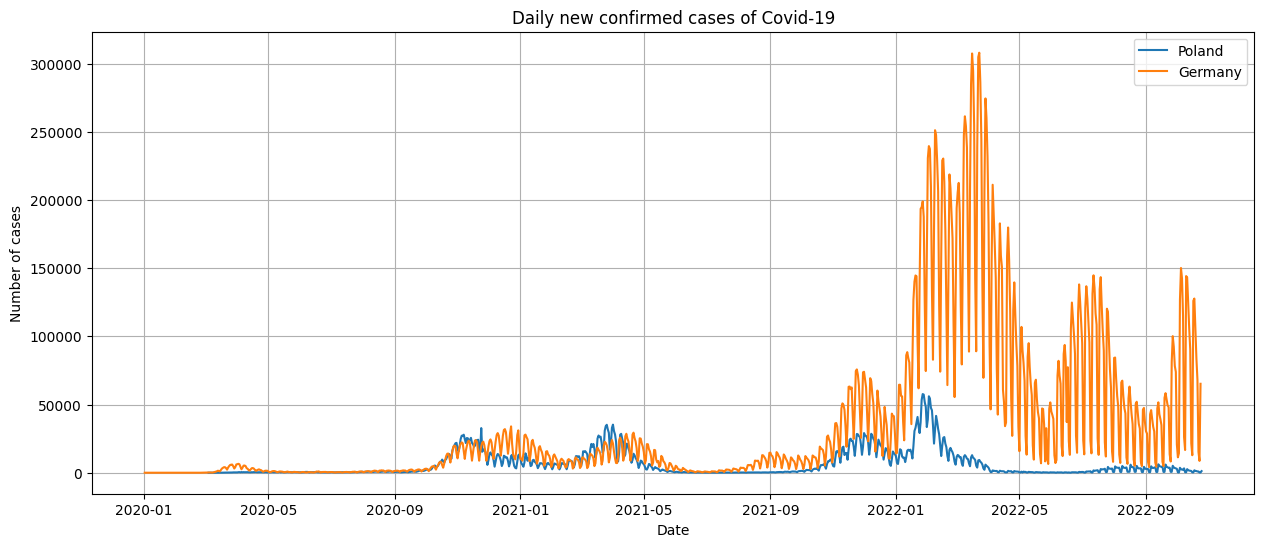
ax.set\_xlabel('Date')

ax.set\_ylabel('Number of cases')

ax.legend()

ax.grid()

plt.show()



*# Find the date with the maximum number of cases for Poland and Germany*

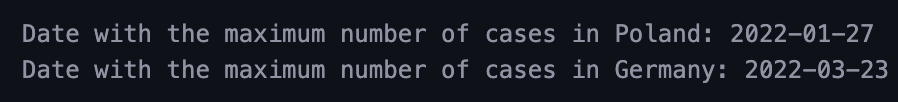
poland\_max\_cases\_date = poland\_covid.loc[poland\_covid['cases'].idxmax(), 'dateRep']

germany\_max\_cases\_date = germany\_covid.loc[germany\_covid['cases'].idxmax(), 'dateRep']

*# Print the results*

print(f"Date with the maximum number of cases in Poland: {poland\_max\_cases\_date.date()}")

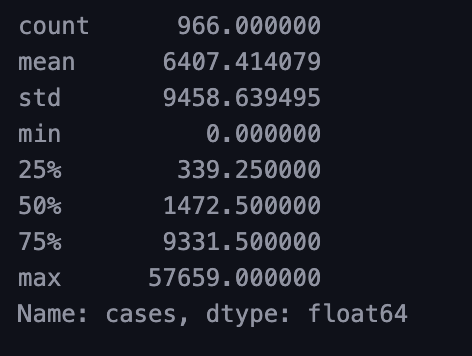
print(f"Date with the maximum number of cases in Germany: {germany\_max\_cases\_date.date()}")



*Дослідимо часові ряди.*

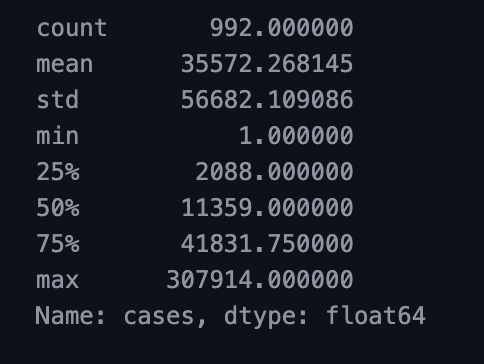
poland\_covid\_c = poland\_covid['cases']

poland\_covid\_c.describe()



germany\_covid\_c = germany\_covid['cases']

germany\_covid\_c.describe()



fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 4))

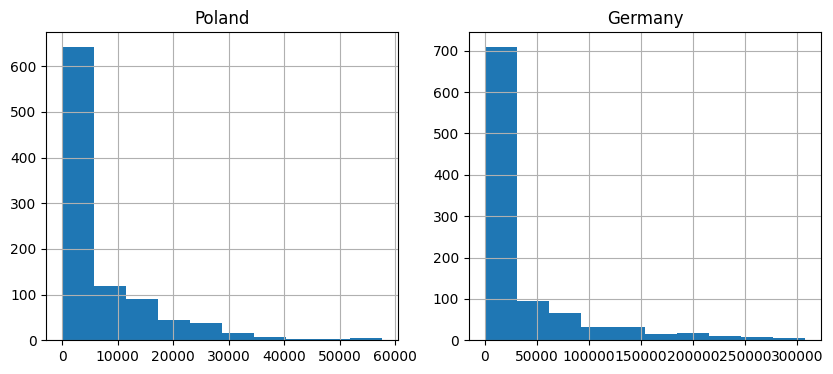
poland\_covid\_c.hist(ax=ax1)

ax1.set\_title('Poland')

germany\_covid\_c.hist(ax=ax2)

ax2.set\_title('Germany')

plt.show()



*Для кращої візуалізації властивостей ряду (трендів, сезонності тощо) застосуємо згладжування за допомогою ковзаючого середнього.*

def plot\_moving\_average(series, n):

rolling\_mean = series['cases'].rolling(window=n).mean()

fig, ax = plt.subplots(figsize=(15, 6))

ax.set\_xlabel('Date')

ax.set\_ylabel('Number of cases')

ax.plot(series['dateRep'], rolling\_mean, c='orange', label='Rolling mean trend')

ax.plot(series['dateRep'], series['cases'], label='Actual values')

ax.legend(loc='upper left')

ax.grid(True)

ax.legend()

ax.grid()

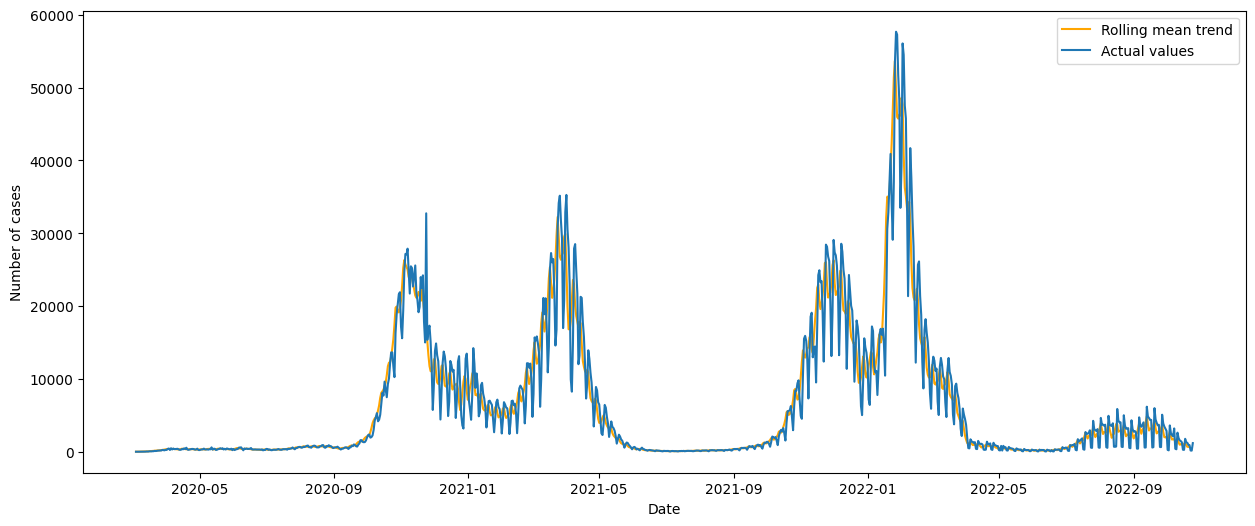
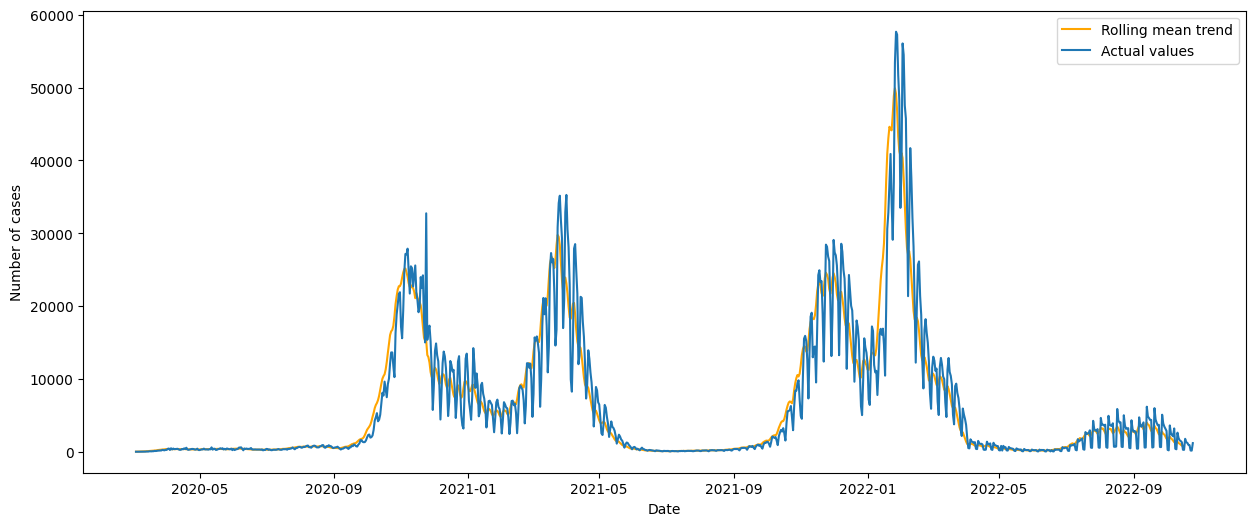
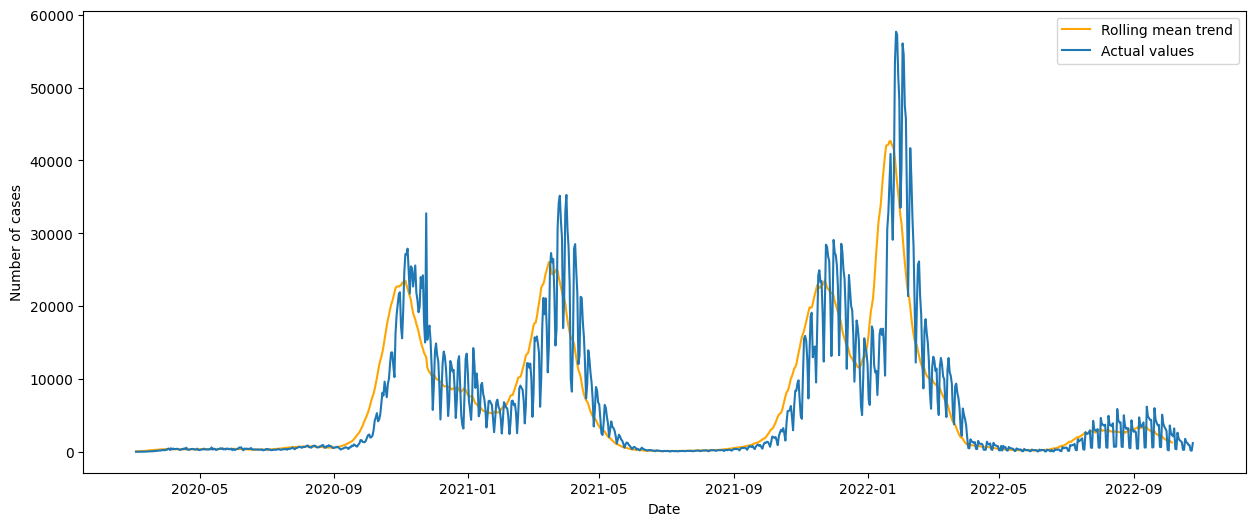
plt.show()

print('Poland: ')

plot\_moving\_average(poland\_covid, 5)

plot\_moving\_average(poland\_covid, 10)

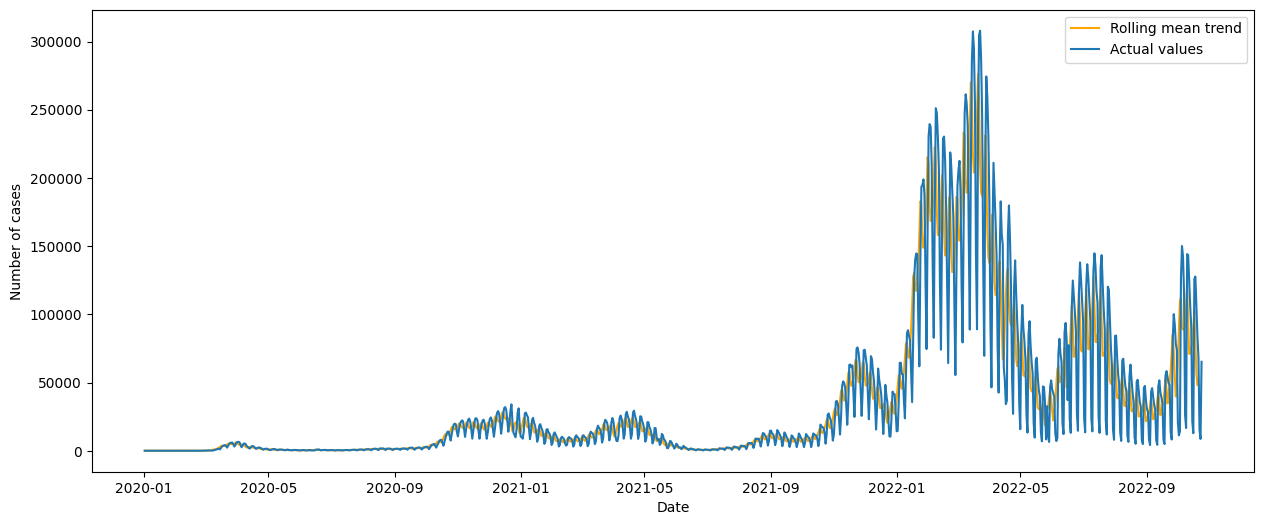
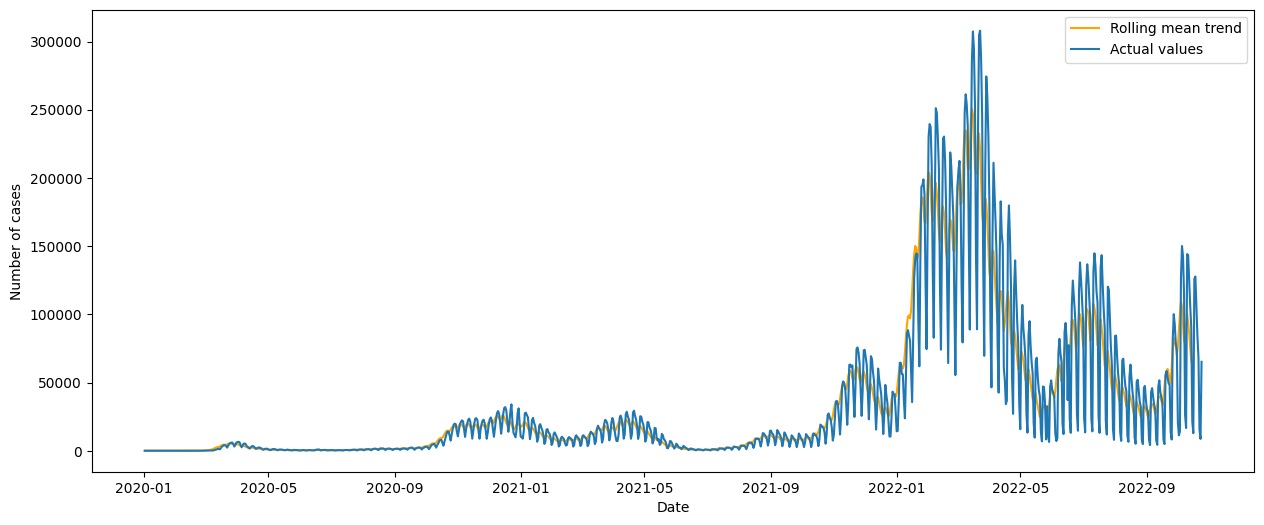
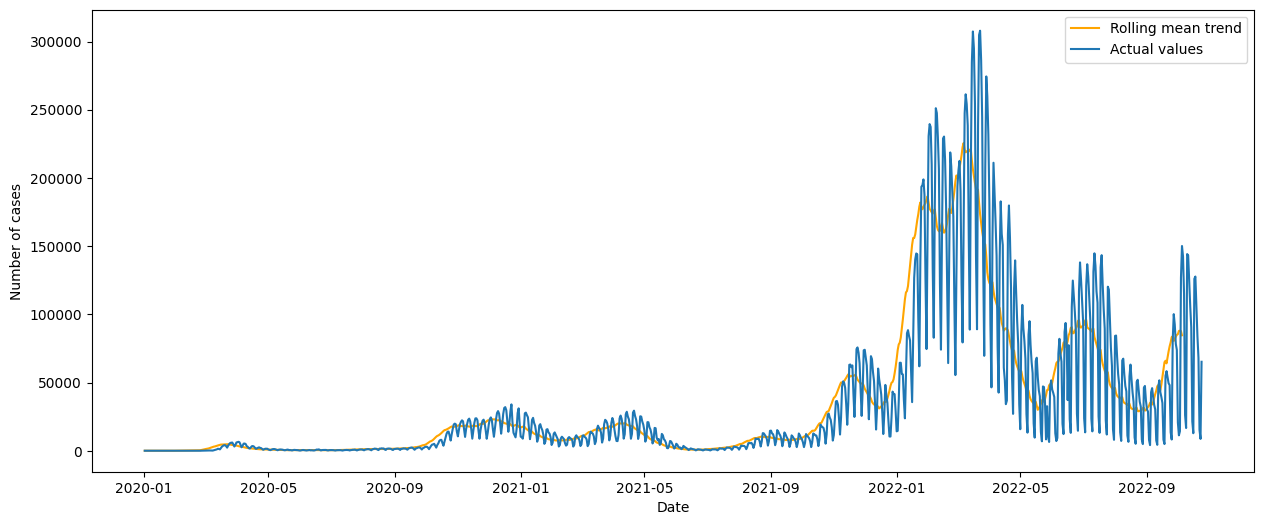
plot\_moving\_average(poland\_covid, 20)

** **  print('Germany: ')

plot\_moving\_average(germany\_covid, 5)

plot\_moving\_average(germany\_covid, 10)

plot\_moving\_average(germany\_covid, 20)

** ** 

*Візуалізуємо декомпозицію ряду на тренд, сезонність та залишки.*

poland\_covid\_n = poland\_covid[['dateRep','cases']]

poland\_covid\_n = poland\_covid\_n.dropna()

poland\_covid\_n = poland\_covid\_n.sort\_values(by='dateRep')

decomposition = smt.seasonal\_decompose(poland\_covid\_n.set\_index('dateRep')['cases'])

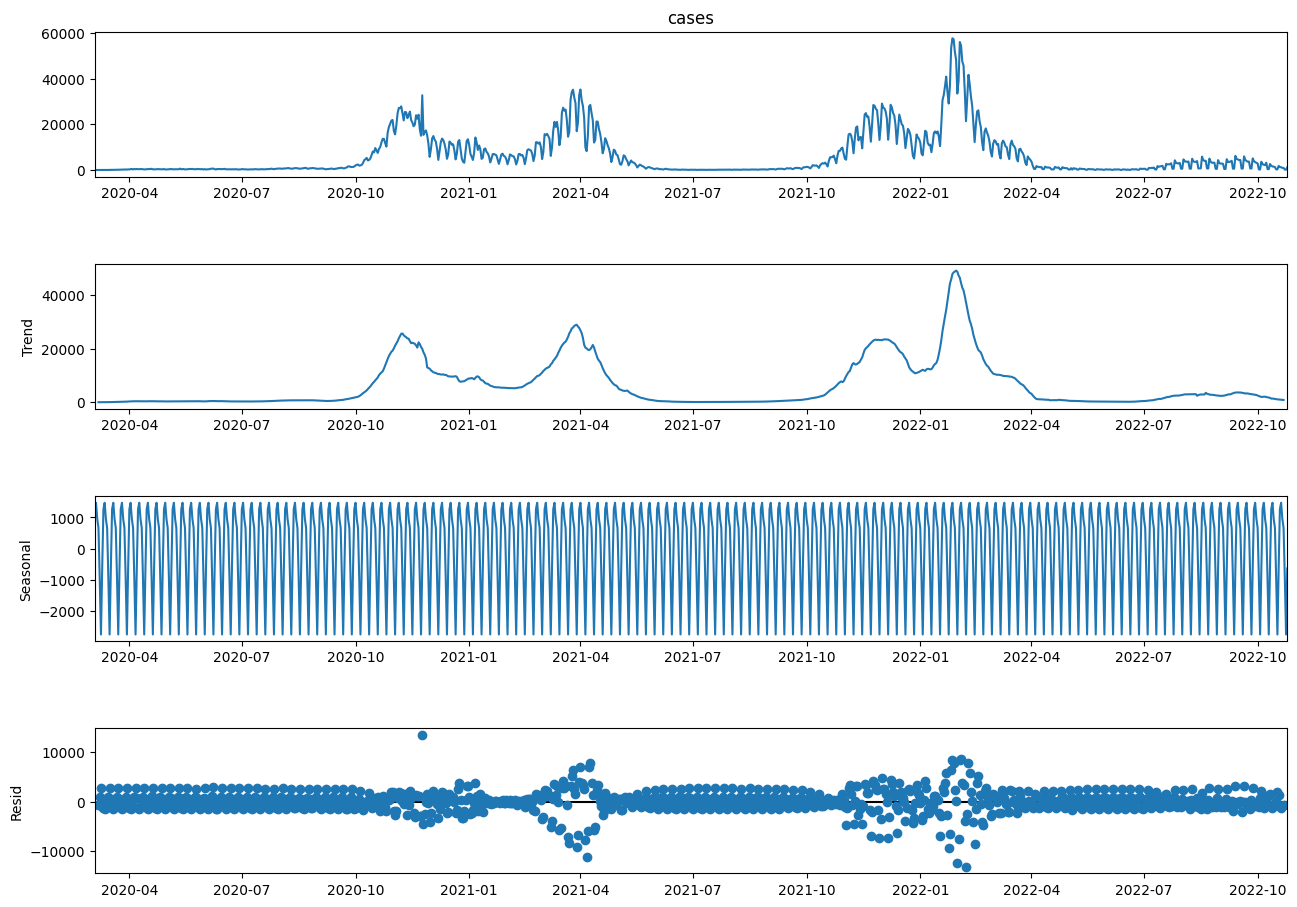
*# Plot the results*

print('Poland: ')

fig = decomposition.plot()

fig.set\_size\_inches(15, 10)

plt.show()



*Будуємо графіки автокореляції та часткової автокореляції.*

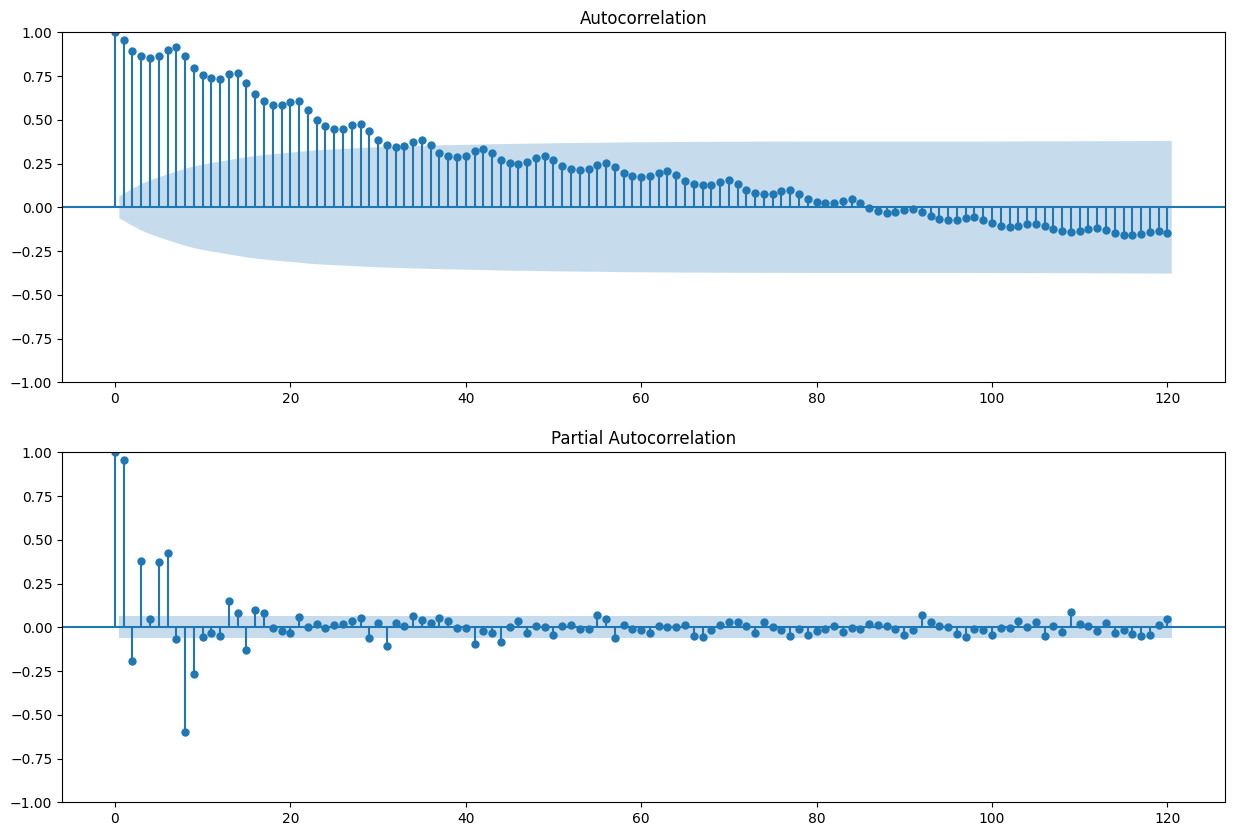
poland\_covid\_n = poland\_covid[['dateRep', 'cases']]

fig, ax = plt.subplots(2, figsize=(15, 10))

ax[0] = plot\_acf(poland\_covid\_n['cases'], ax=ax[0], lags=120)

ax[1] = plot\_pacf(poland\_covid\_n['cases'], ax=ax[1], lags=120)

plt.show()



*Перевіримо ряд на стаціонарність за допомогою доповненого тесту Дікі-Фуллера.*

def dickey\_fuller\_test(series):

test = smt.adfuller(series, autolag='AIC')

print('adf: ', test[0])

print('p-value: ', test[1])

print('Critical values: ', test[4])

*if* test[0] > test[4]['5%']:

print(' There are unit roots, the series is not stationary.')

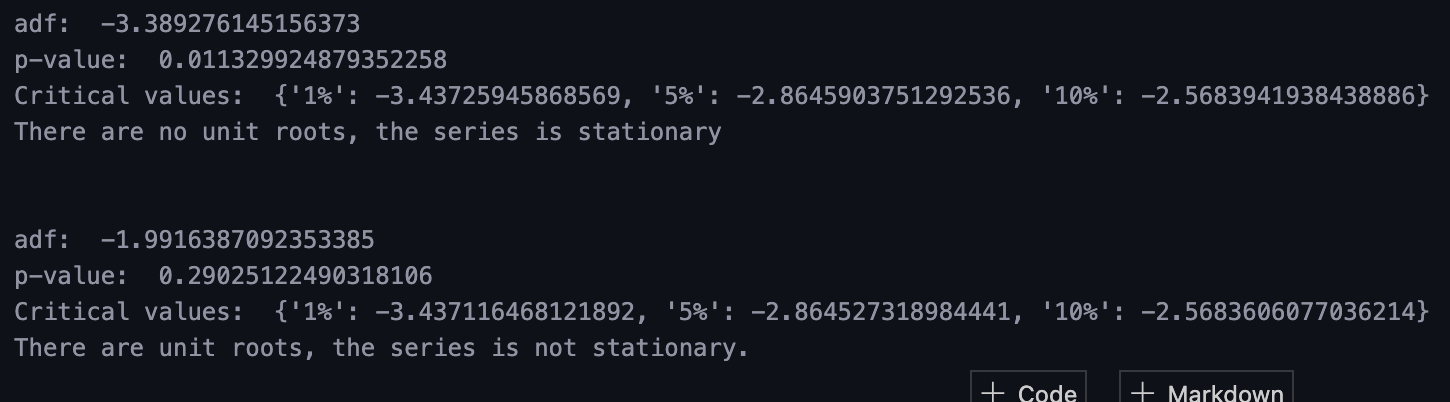
*else*:

print(' There are no unit roots, the series is stationary.')

dickey\_fuller\_test(poland\_covid\_n['cases'])

print('\n')

dickey\_fuller\_test(germany\_covid['cases'])



*Зчитаємо файл*

df = pd.read\_csv('uah-usd.csv', sep=',', decimal=',', encoding='windows-1251')

*# Convert data types to the appropriate types*

df['Date'] = pd.to\_datetime(df['Date'], format='%Y-%m-%d')

df[['Open', 'High', 'Low', 'Close', 'Adj Close']] = df[['Open', 'High', 'Low', 'Close', 'Adj Close']].astype(float)

*# Sort the dataframe by date*

df.sort\_values(by='Date', inplace=True)

df.set\_index('Date', inplace=True)

df.index.freq = pd.date\_range(start=df.index[0], end=df.index[-1], periods=len(df)).inferred\_freq

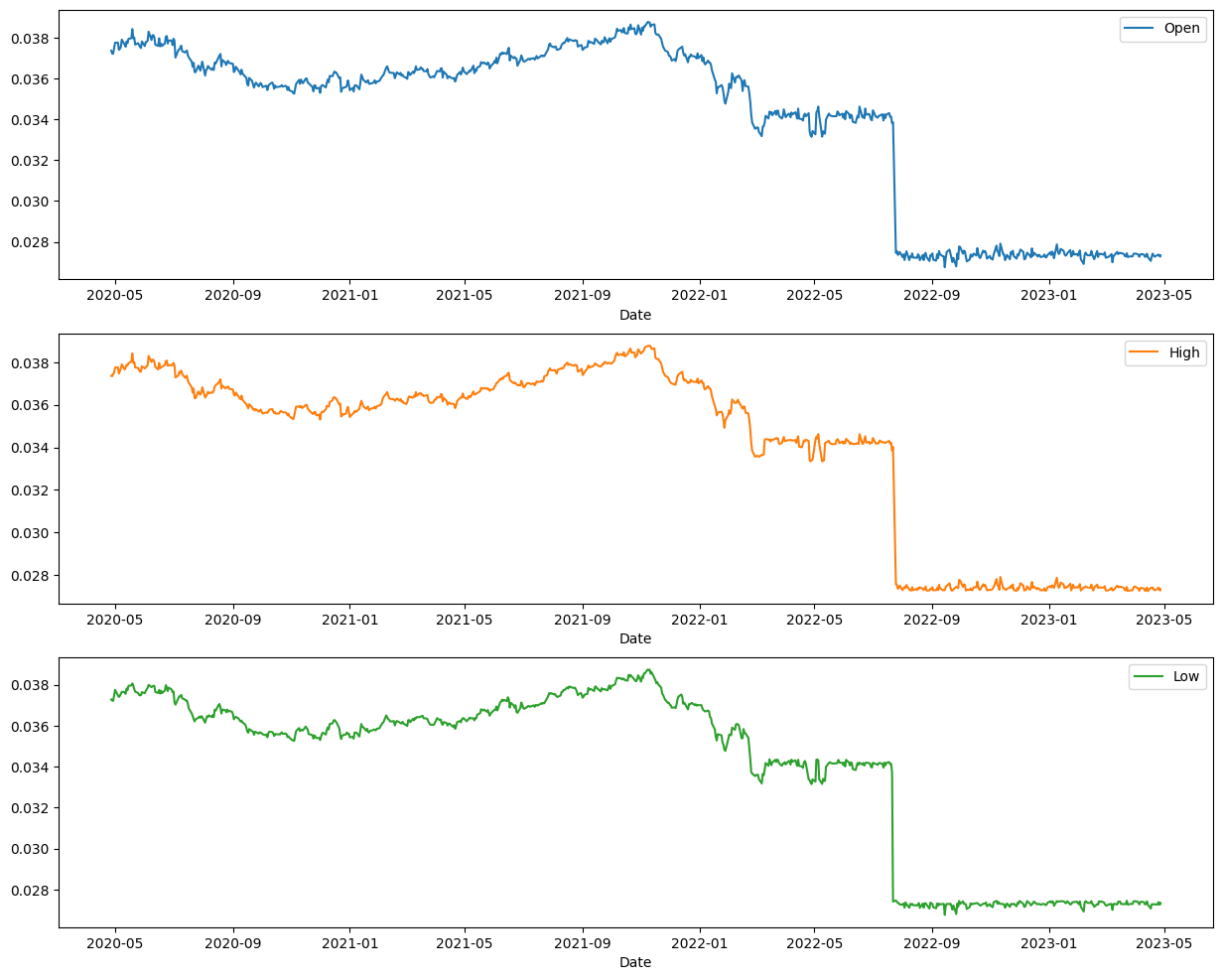
*Будуємо динаміку ціни, найвищої та найнижчої за добу*

fig, ax = plt.subplots(figsize=(15, 12))

df[['Open', 'High', 'Low']].plot(ax=ax, subplots=True)

ax.grid()

plt.show()



*Використовуємо згладжування для дослідження характеристик рядів*

def moving\_average(series, n):

rolling\_mean = series.rolling(window=n).mean()

plt.figure(figsize=(15, 5))

plt.plot(rolling\_mean, c='orange', label='Rolling mean trend')

plt.plot(series[n:], label='Actual values')

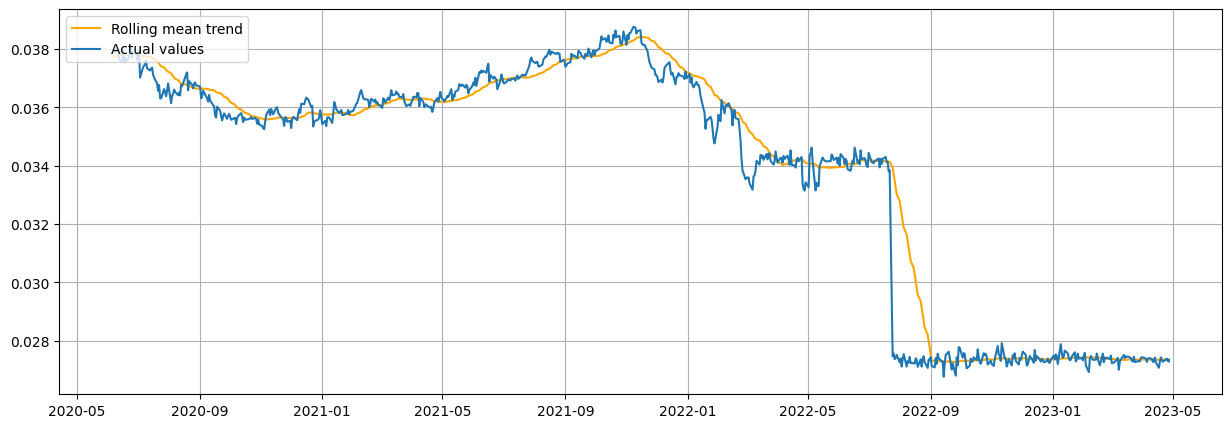
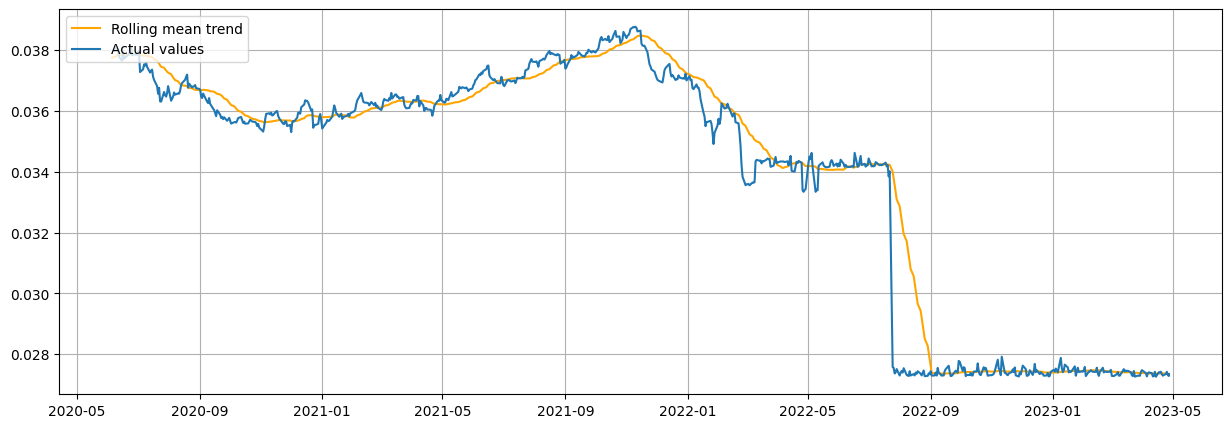
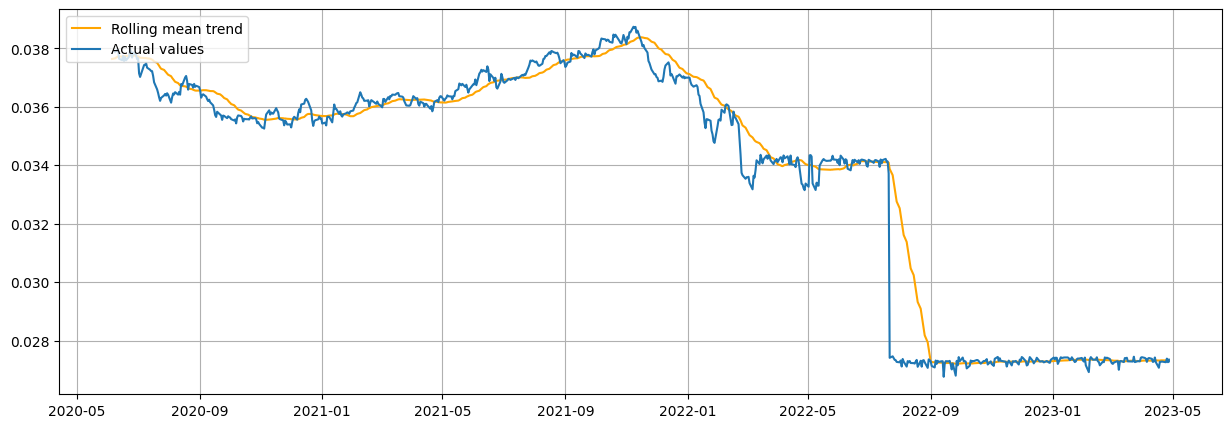
plt.legend(loc='upper left')

plt.grid(True)

 moving\_average(df['Open'], 30)

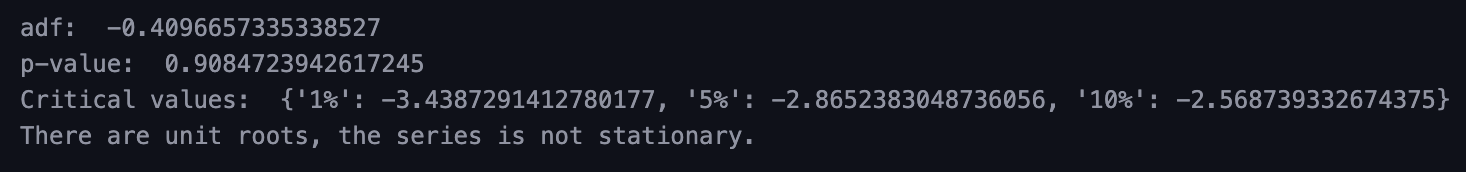
moving\_average(df['High'], 30)

moving\_average(df['Low'], 30)

*Перевіримо ряд на стаціонарність за допомогою доповненого тесту Дікі-Фуллера.*

dickey\_fuller\_test(df['Open'])



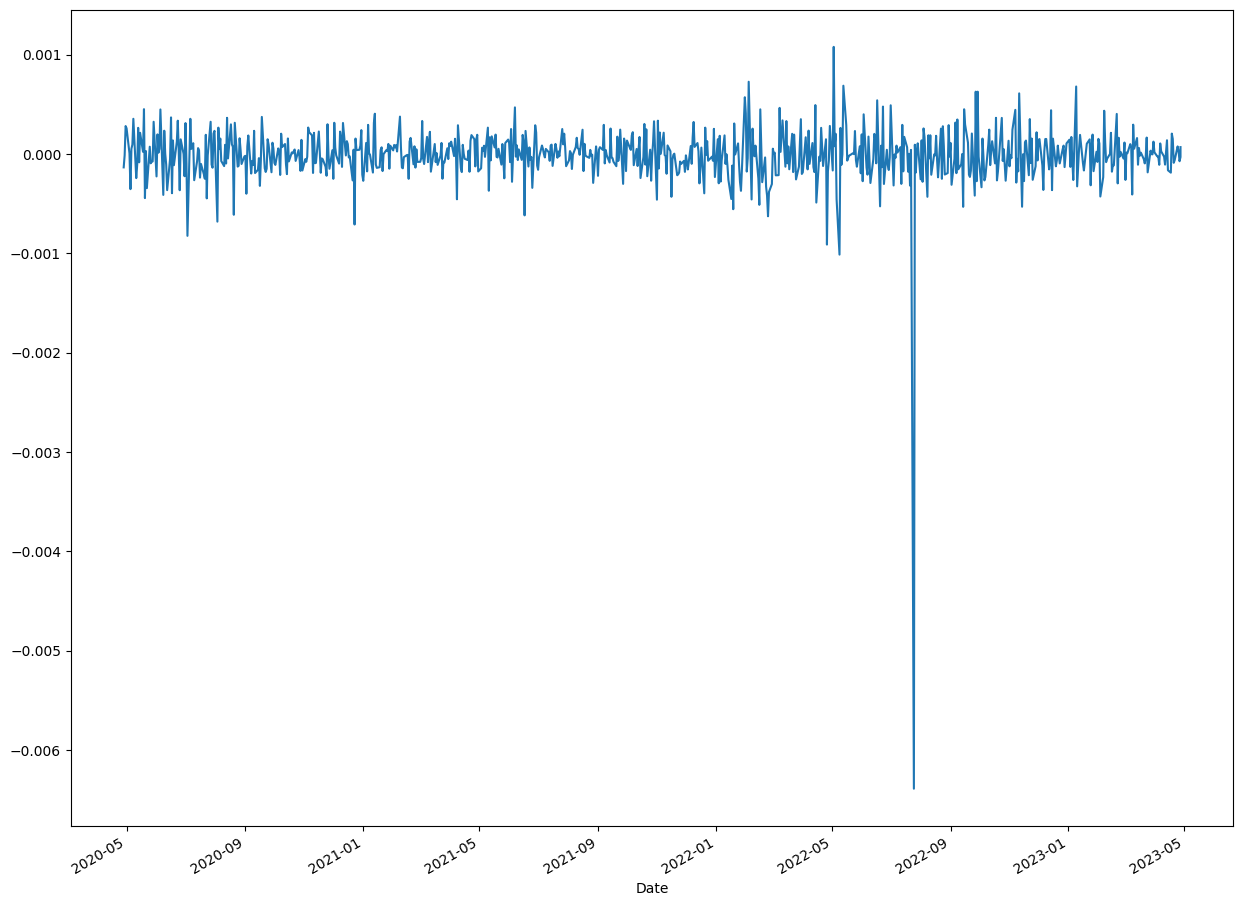
*Зробимо стаціонарним*

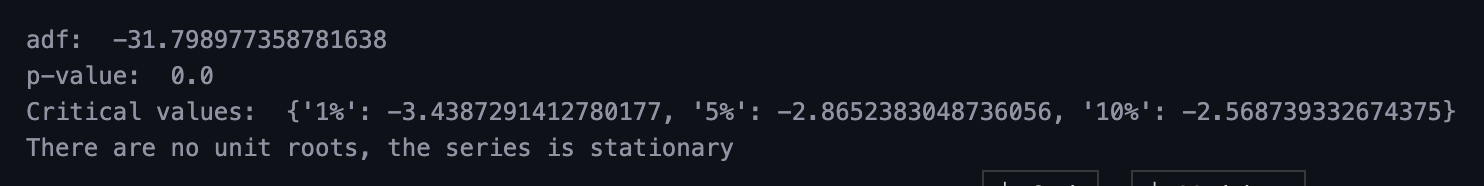
currencies\_price\_df\_diff = df['Open'].diff(periods=1).dropna()

fig, ax = plt.subplots(figsize=(15, 12))

currencies\_price\_df\_diff.plot(ax=ax)

plt.show()

 dickey\_fuller\_test(currencies\_price\_df\_diff)

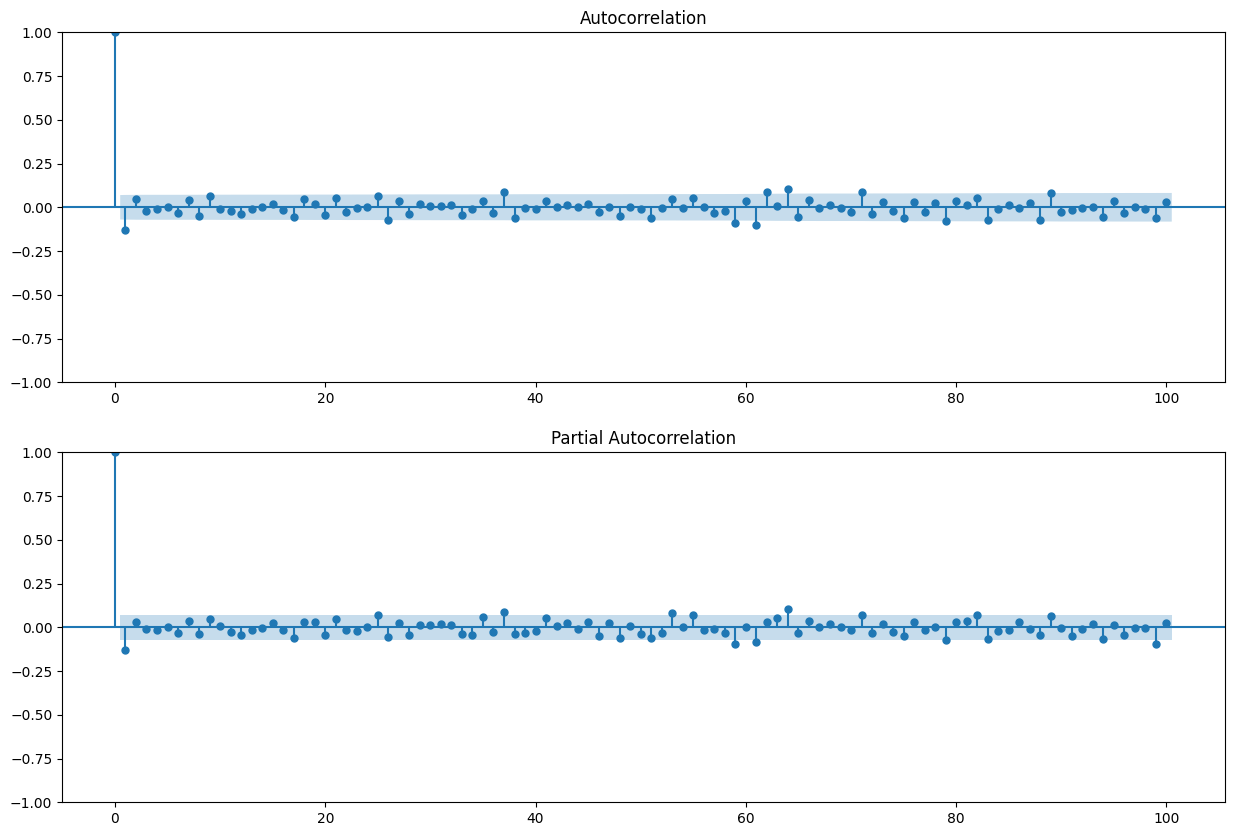


*Будуємо графіки автокореляції та часткової автокореляції*

fig, ax = plt.subplots(2, figsize=(15, 10))

ax[0] = plot\_acf(currencies\_price\_df\_diff, ax=ax[0], lags=100)

ax[1] = plot\_pacf(currencies\_price\_df\_diff, ax=ax[1], lags=100)

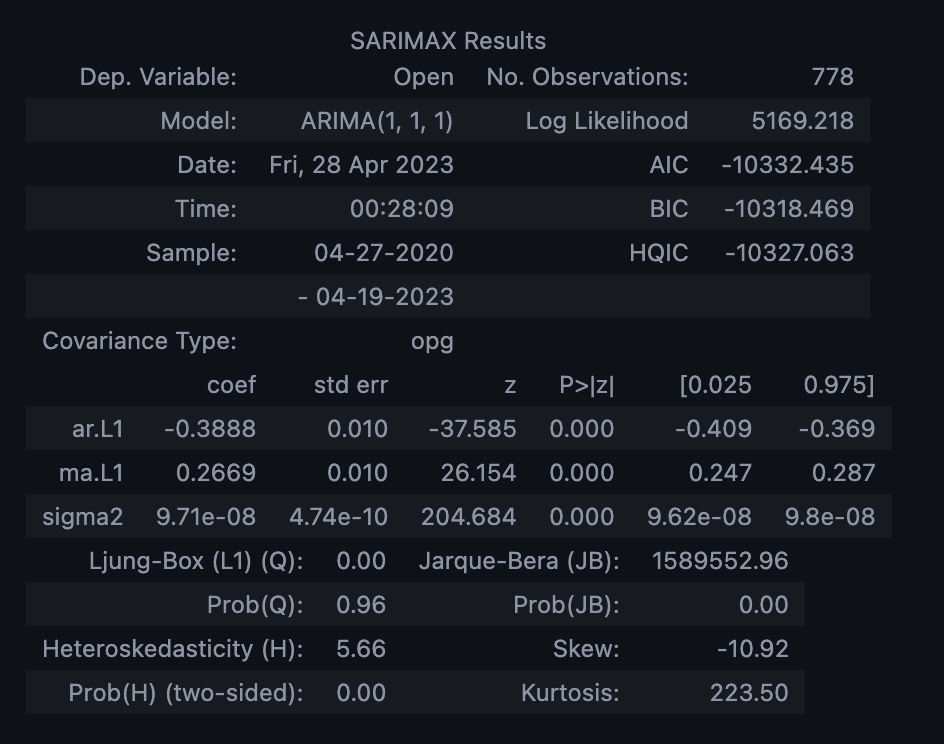


*Будуємо модель ARIMA для прогнозу значення ціни на тиждень вперед*

train\_data = df['Open'][:-7]

model = smt.ARIMA(train\_data, order=(1, 1, 1)).fit()

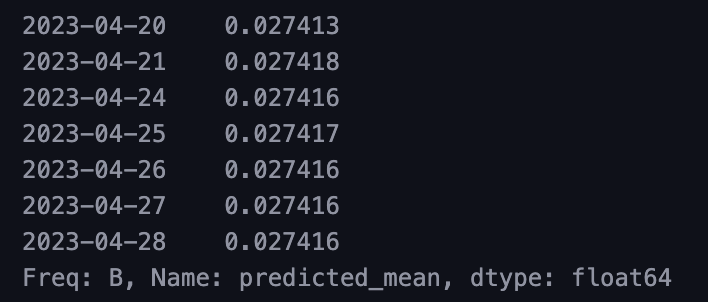
model.summary()



pred = model.predict(df['Open'].index[-7], df['Open'].index[-1])

test\_data = currencies\_price\_df\_diff[-7:]

forecasts = model.forecast(7)

forecasts  


**Додаткове завдання:**

*Імпортуємо потрібні бібліотеки.*

*import* numpy *as* np

*import* pandas *as* pd

*import* matplotlib.pyplot *as* plt

*import* seaborn *as* sns

*import* statsmodels.api *as* sm

*import* statsmodels.tsa.api *as* smt

*from* statsmodels.graphics.tsaplots *import* plot\_acf, plot\_pacf

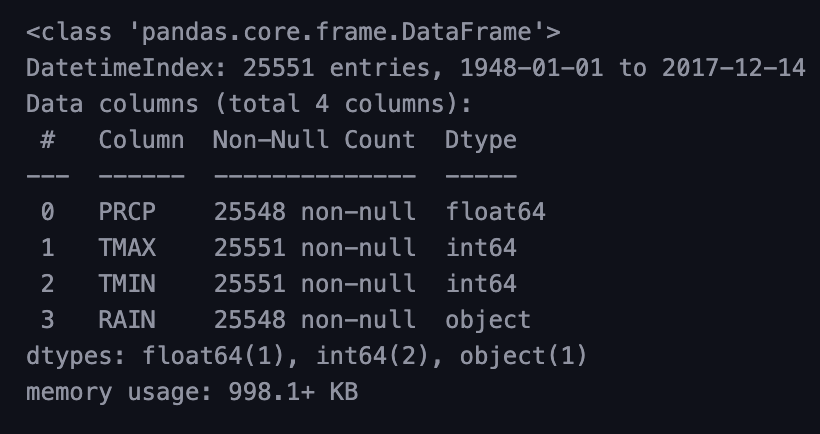
*from* statsmodels.graphics.tsaplots *import* plot\_predict

*Завантажемо файл.*

df = pd.read\_csv('seattleWeather\_1948-2017.csv', index\_col=['DATE'], parse\_dates=['DATE'])

*Дослідимо дані*

df.info()

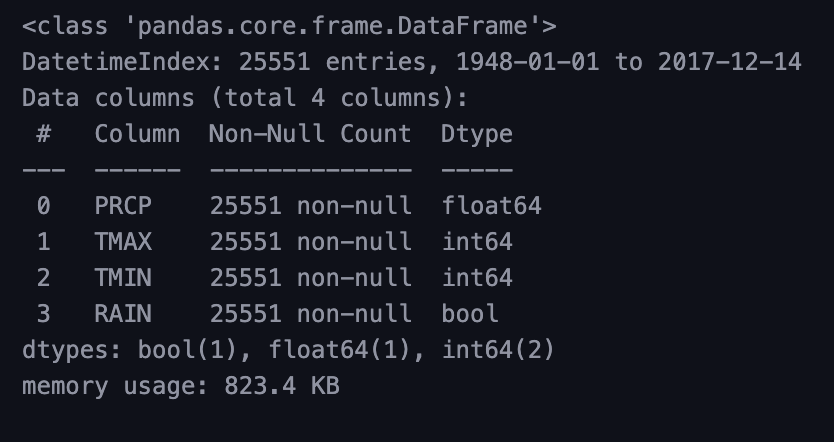


*Заповнюємо пропущені значення*

df['PRCP'].fillna(0, inplace=True)

df['RAIN'].fillna(False, inplace=True)

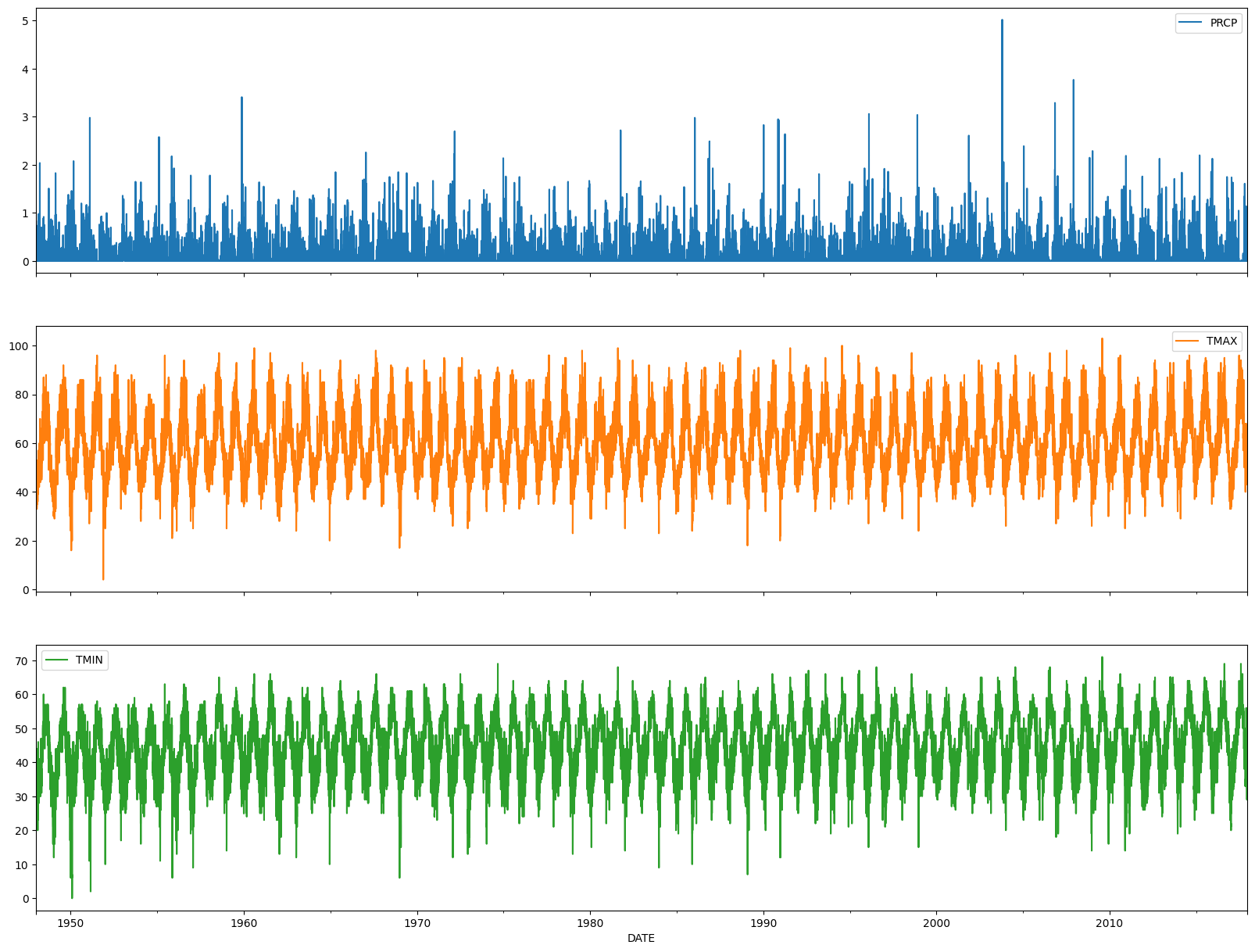
df.info()



*Будуємо динаміку опадів та температур в часі*

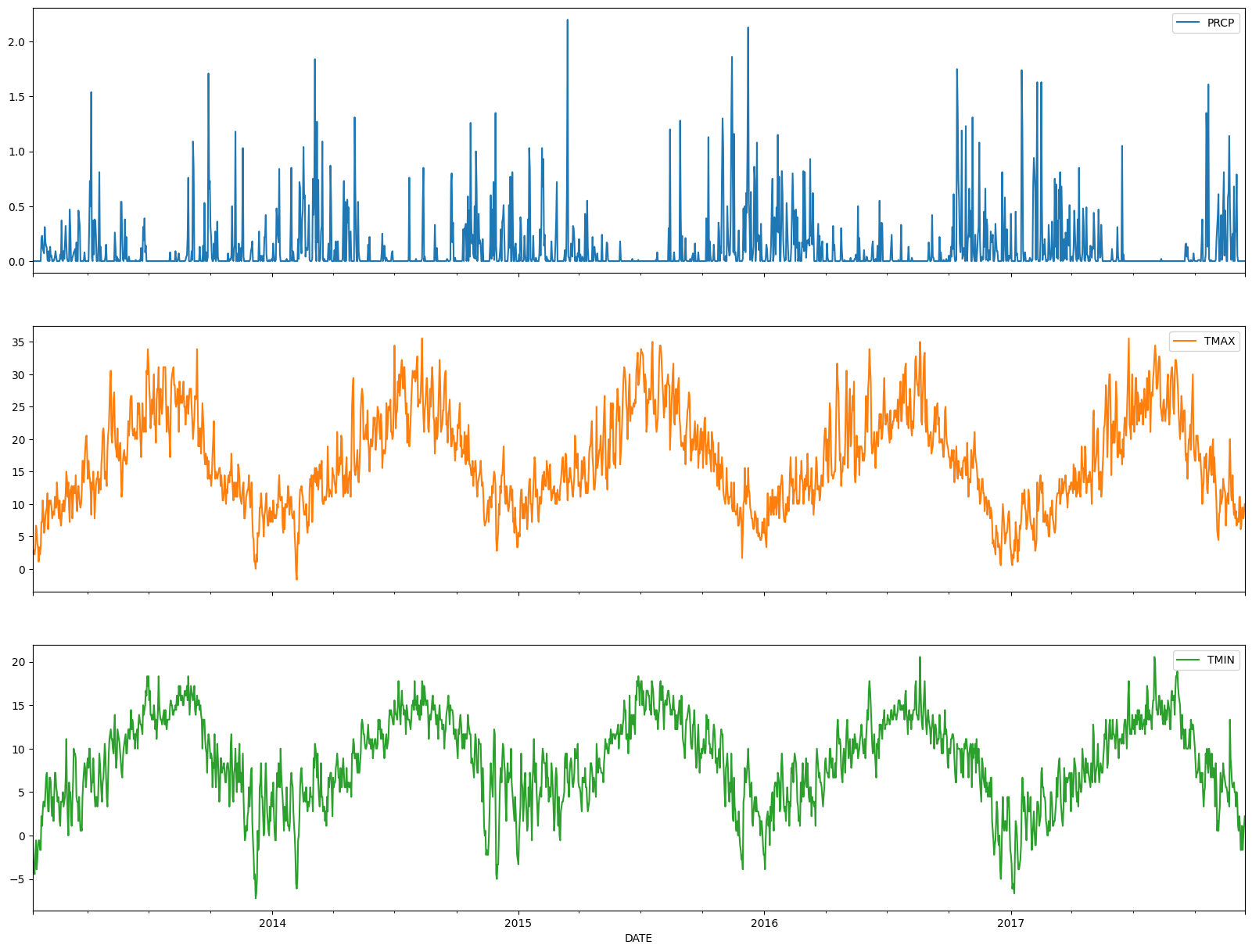
df.plot(figsize=(20, 15), subplots=True)

plt.show()



df.loc[df.index[-1800:]].plot(figsize=(20, 15), subplots=True)

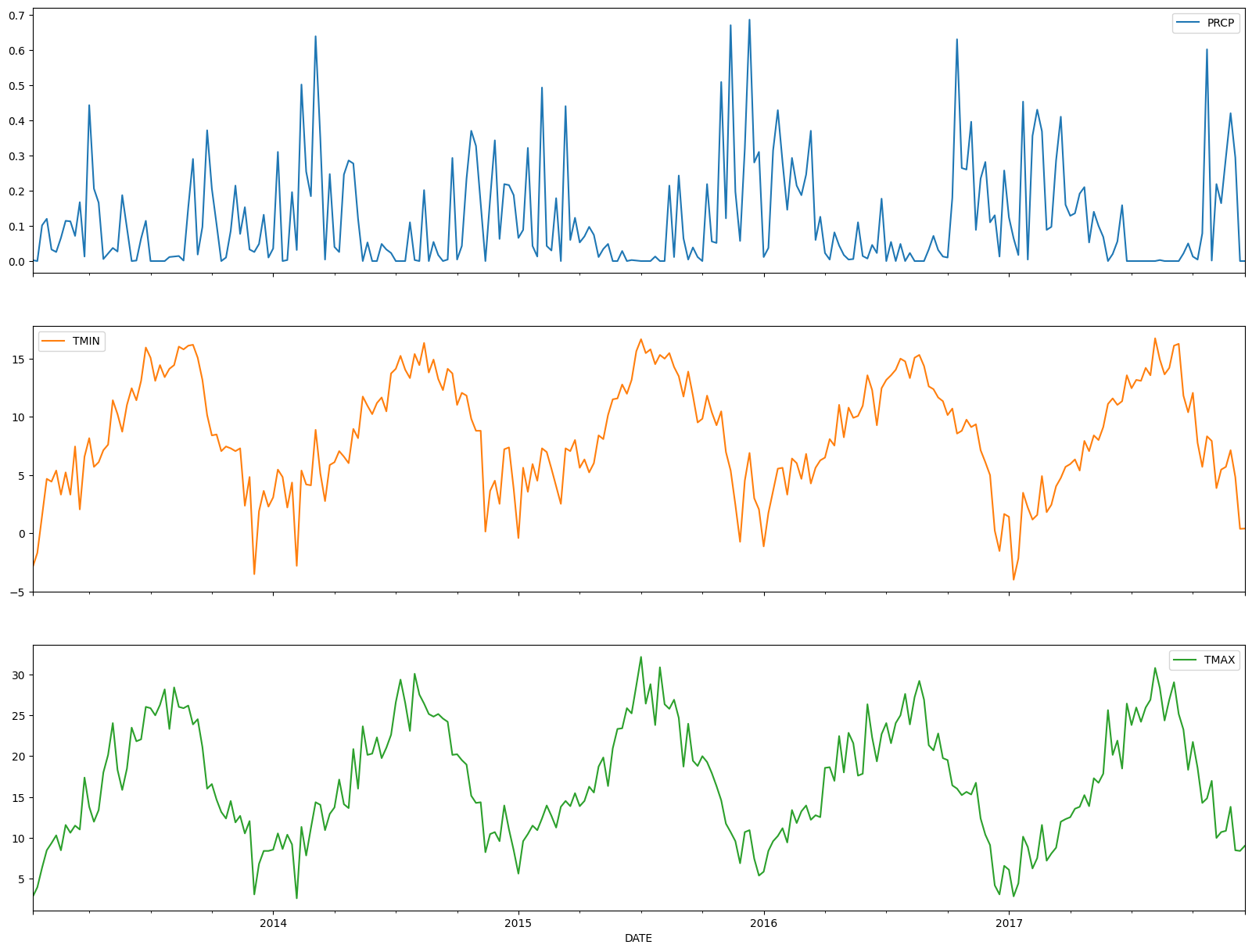
plt.show()



*Динаміка для середніх значень показників по тижням*

df[['PRCP', 'TMIN', 'TMAX']].loc[df.index[-1800:]].resample('W').mean().plot(figsize=(20, 15), subplots=True)

plt.show()



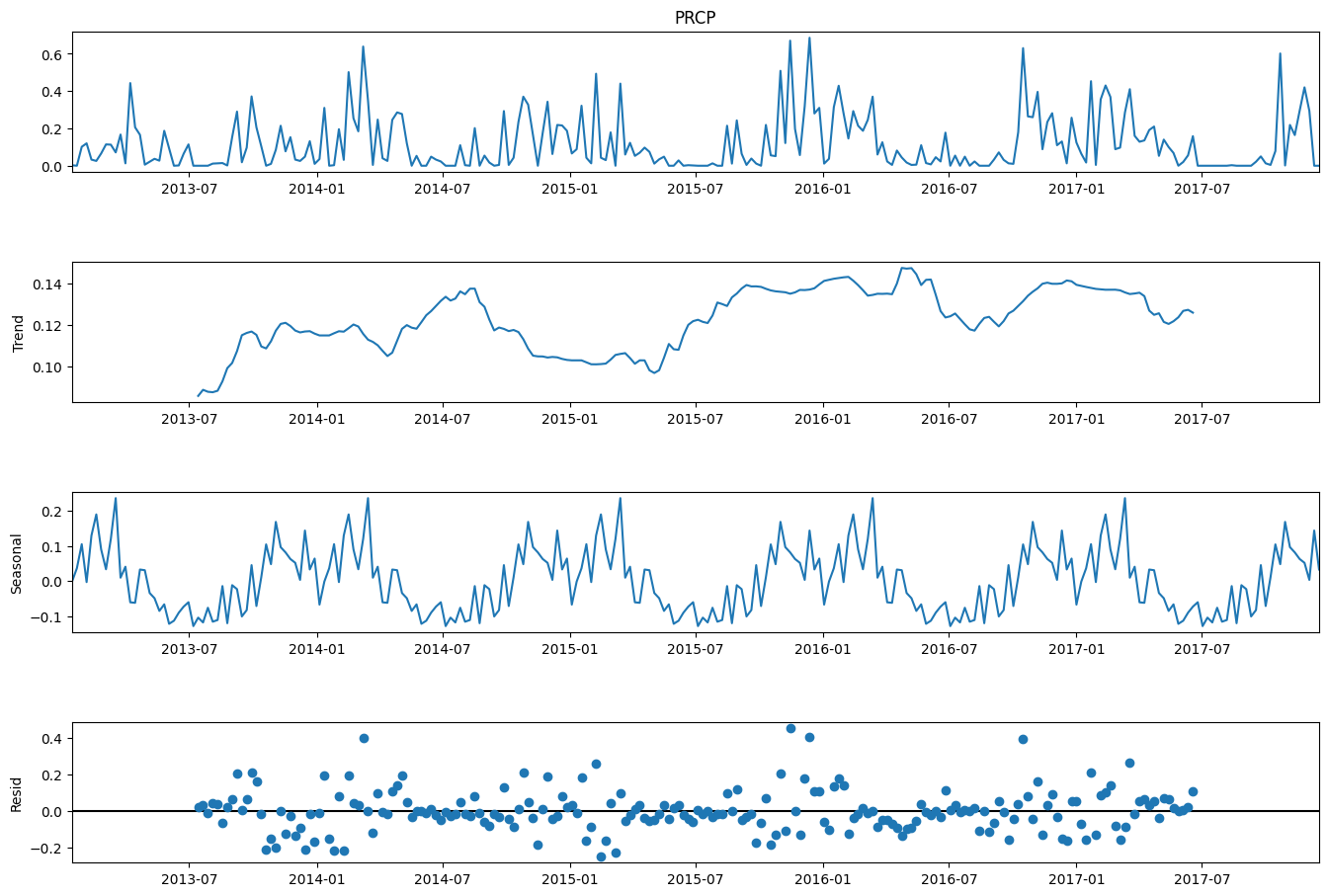
*Декомпозуємо*

prcp\_decomposition = smt.seasonal\_decompose(df['PRCP'].loc[df.index[-1800:]].resample('W').mean())

fig = prcp\_decomposition.plot()

fig.set\_size\_inches(15, 10)

plt.show()

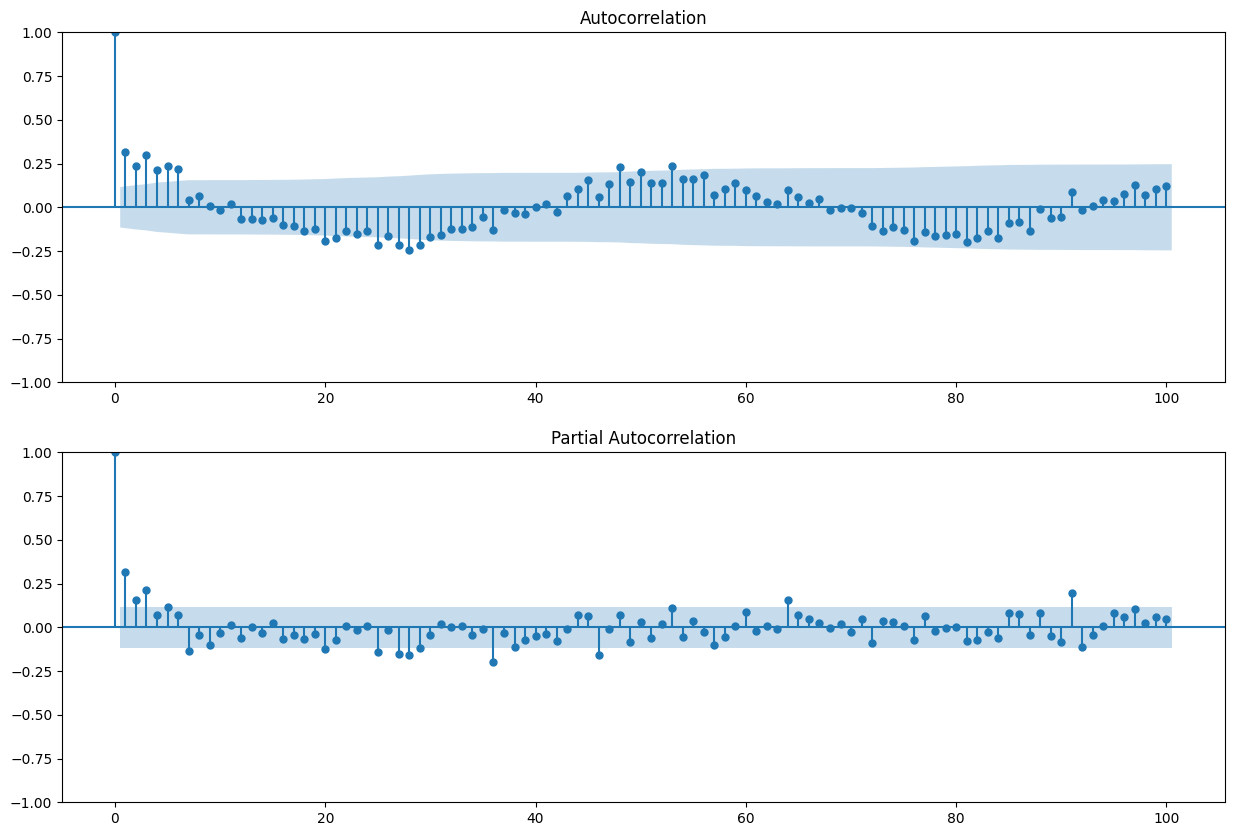


*Аутокореляція*

fig, ax = plt.subplots(2, figsize=(15, 10))

ax[0] = plot\_acf(df['PRCP'].loc[df.index[-2000:]].resample('W').mean(), ax=ax[0], lags=100)

ax[1] = plot\_pacf(df['PRCP'].loc[df.index[-2000:]].resample('W').mean(), ax=ax[1], lags=100)



*Фаренгейт у Цульсії*

df[['TMIN', 'TMAX']] = (df[['TMIN', 'TMAX']] - 32) \* 5 / 9

df[['TMIN', 'TMAX']].describe()



*Будуємо матрицю кореляцій*

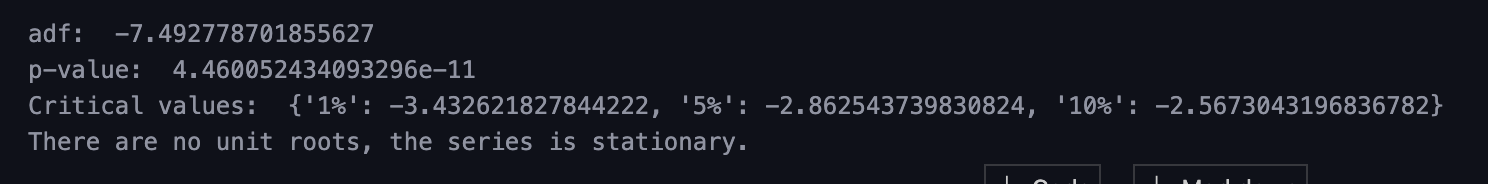
corrmat = df[['PRCP', 'TMIN', 'TMAX']].corr()

corrmat



*Діккі-Фюллєр*

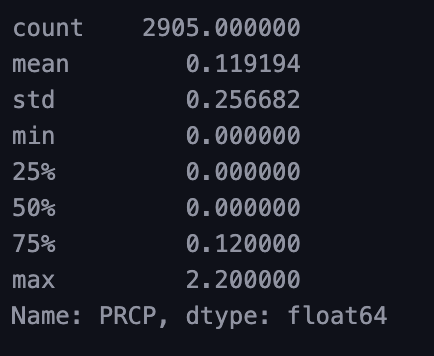
dickey\_fuller\_test(df['PRCP'].loc['2010-01-01':])



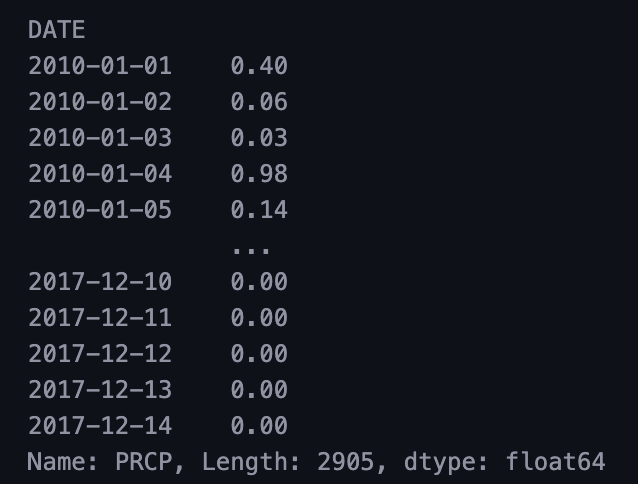
*Будуємо модель ARIMA для передбачення опадів*

train\_data = df['PRCP'].loc['2010-01-01':]

train\_data.describe()

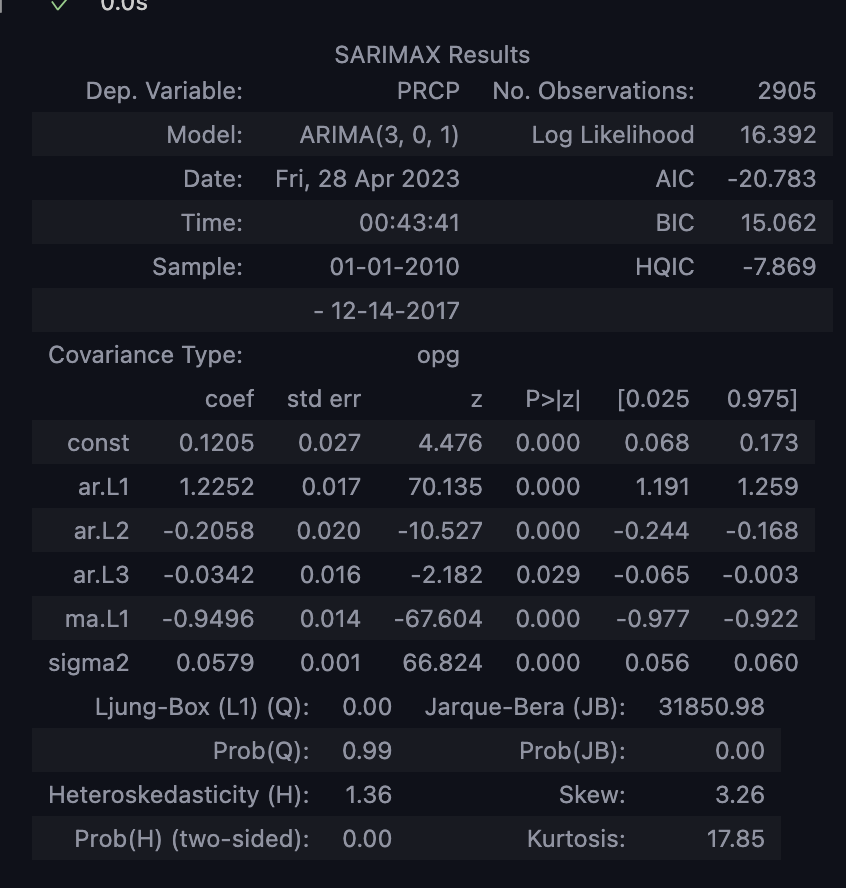


train\_data



model = smt.ARIMA(train\_data, order=(3, 0, 1)).fit()

model.summary()



fig, ax = plt.subplots(figsize=(15, 10))

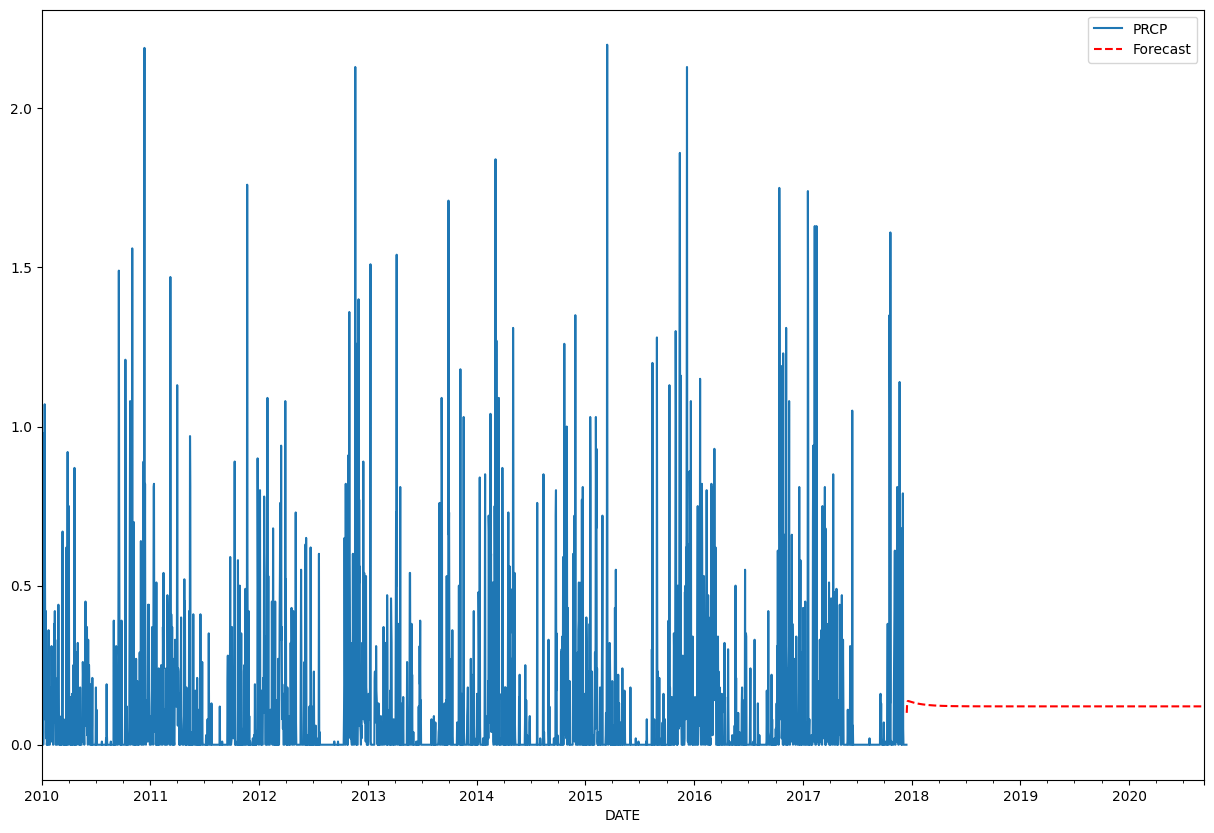
ax = train\_data.plot(ax=ax)

forecast = model.forecast(steps=1000)

forecast.plot(ax=ax, style='r--', label='Forecast')

plt.legend()

plt.show()



Висновок

За отриманими даними можна зробити висновок, що

* В основному завданні було проаналізовано пандемію у Польщі та Німеччині, де помітно час хвиль вірусу. Було проаналізовано гривню до долара. Долар є однію зі стійкіших валют, при тому гривня постійно приймали на себе катаклізми та тільки спадала, але не залежно від прогнозу точно зросте.
* В додатковому завданні був зроблений прогноз на 1000 шагів з 2018 року.