|  |  |  |
| --- | --- | --- |
| Indeks |  |  |
| Grupa |  | |

W miarę możliwości/potrzeby proszę o dodatkowy komentarz do kodu – w szczególności, warto zaznaczyć zrealizowane dodatkowe elementy, czy skomentować przyjętą interpretację pewnych elementów składowych.

Celem zadań jest zapoznanie się z możliwościami rozszerzenia funkcjonalności Power BI Desktop o własne skrypty przygotowane w języku Python – zarówno w warstwie dostępu do danych, warstwie przetwarzania danych oraz warstwie wizualizacji danych.

Korzystając z danych z zewnętrznego API (NBP) chcemy uzupełnić model z poprzedniej listy zadań o możliwość wyrażenia wartości sprzedaży (w firmie AW) w PLN (wartości sprzedaży wyznaczone są w walucie USD, a chcielibyśmy je przeliczyć na PLN – interesują nas jedynie kwoty zamówień).

Wszystkie zadania zrealizuj w ramach jednego pliku .pbix.

# Zadanie 1

Jako rozwiązanie zadania prześlij kod źródłowy przygotowanej funkcji oraz zrzut ekranu przedstawiający wykres zmiany kursu walut.

## Funkcja

## Wykres

## Konkluzje dot. zadaniA 1

# Zadanie 2

Jako rozwiązanie zadania prześlij kod zapytania zawierającego wszystkie daty oraz skrypt Python.

## KOD – POWER QUERY – Zapytania

let

Source = Table.FromRows(Json.Document(Binary.Decompress(Binary.FromText("i44FAA==", BinaryEncoding.Base64), Compression.Deflate)), let \_t = ((type nullable text) meta [Serialized.Text = true]) in type table [Column1 = \_t]),

#"Changed Type" = Table.TransformColumnTypes(Source,{{"Column1", type text}}),

#"Run Python script" = Python.Execute("# Python:#(lf)import pandas as pd#(lf)df3 = pd.merge(df1, df2, how = 'right', on = ['effectiveDate'])#(lf)df3 = df3.sort\_values(by=['effectiveDate'])#(lf)df3 = df3.fillna(method='ffill')",[df1=rates, df2=dates]),

df3 = #"Run Python script"{[Name="df3"]}[Value],

#"Changed Type1" = Table.TransformColumnTypes(df3,{{"currency", type text}, {"code", type text}, {"no", type text}, {"effectiveDate", type date}, {"mid", type text}}),

#"Sorted Rows" = Table.Sort(#"Changed Type1",{{"effectiveDate", Order.Ascending}}),

#"Changed Type2" = Table.TransformColumnTypes(#"Sorted Rows",{{"effectiveDate", type text}}),

#"Sorted Rows1" = Table.Sort(#"Changed Type2",{{"effectiveDate", Order.Descending}}),

#"Changed Type3" = Table.TransformColumnTypes(#"Sorted Rows1",{{"effectiveDate", type date}}),

#"Sorted Rows2" = Table.Sort(#"Changed Type3",{{"effectiveDate", Order.Ascending}})

in

#"Sorted Rows2"

## Kod – PYTHON

import pandas as pd

import requests

import json

def import\_data(curr, dates):

start, end = dates

url = f'http://api.nbp.pl/api/exchangerates/rates/a/{curr}/{start}/{end}/'

response\_api = requests.get(url)

data = response\_api.text.encode().decode('utf-8-sig')

parse\_json = json.loads(data)

df = pd.DataFrame(parse\_json)

return df

def clean\_data(df):

df['rates'].apply(pd.Series)

df = pd.concat([df, df['rates'].apply(pd.Series)], axis=1)

df.drop(columns=['rates','table'], inplace=True)

return df

currency = 'usd'

daterange = ('2022-01-01', '2023-01-01')

data = import\_data(currency, daterange)

rates = clean\_data(data)

dates = pd.date\_range(start='2022-03-01', end='2023-01-01')

dates = pd.to\_datetime(dates, format='%Y-%d-%m')

dates = pd.DataFrame(dates)

dates=dates.astype(str)

df3 = pd.merge(rates, dates, how = 'right', left\_on = ['effectiveDate'], right\_on=0)

df3.fillna(method='ffill')

print(df3)

## Konkluzje dot. zadaniA 2

W skrypcie do zadania 2 użyte zostały dwie zaimplementowane funkcje: import\_data służaca do pobierania danych ich zapisu przy użyciu biblioteki pandas, oraz cleandata() służąca do….

W celu rozwiązania problemu związanego z brakującymi datami dodaliśmy nową tabelę dates Ustawiony został przedział interesujących nas dat – daterange. Nastepnie zmieniony został typ danych w kolumnie dates na string w celu …. Za pomocą funkcji z biblioteki pandas pd.merge() złączone zostały tabele „rates” oraz „dates” prawostronnie aby w wyniku złączenia w kolumnie effectiveDates wyświetlone były wszystkie daty i na koniec za pomocą funkcji fillna() puste rekordy zostały uzupełnione o wartości kursów z dni poprzednich dzięki wartości method = ‘ffill”.

W wyniku tych działań otrzymaliśmy tabele fixed\_rates z rozwiązanym problemem dotyczącym brakujących dat.

# Zadanie 3

Jako rozwiązanie zadania prześlij kod źródłowy Python oraz zrzut ekranu gotowego raportu (wizualizacje).

## KOD – PYTHON

# The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:

# dataset = pandas.DataFrame(effectiveDate, mid)

# dataset = dataset.drop\_duplicates()

# Paste or type your script code here:

import seaborn as sb

import matplotlib.pyplot as plt

sb.histplot(dataset.mid)

plt.show()

## Raport – zrzut ekranu

Obraz zawierający wykres

Opis wygenerowany automatycznie

## Konkluzje dot. zadaniA 3

Do zrobienia zadania 3 zostały użyte zapytania z poprzedniego zadania, natomiast do wykonania wizualizacji została użyta funkcja histplot z biblioteki seaborn, która przyjmuje jeden argument – w tym wypadku jest to atrybut mid z naszych danych który odpowiada za cenę kursu dolara. Wygenerowany histogram przedstawia ile razy pojawiła się dana cena kursu dolara.

### PODSUMOWANIE – Konkluzje do całej listy zadań