import dash

import dash\_core\_components as dcc

import dash\_html\_components as html

import dash\_bootstrap\_components as dbc

from dash.dependencies import Input, Output, State

from dash.dash import no\_update

from dash\_extensions import Download

from dash\_extensions.snippets import send\_data\_frame

from dash\_extensions.snippets import send\_file

from flask import Flask

from waitress import serve

import dash\_table

import numpy as np

import pandas as pd

from plotly.subplots import make\_subplots

import plotly.graph\_objects as go

from sklearn.ensemble import RandomForestRegressor

import joblib

import plotly.express as px

import base64

#from datetime import date

import openpyxl as opxl

from openpyxl.utils.cell import get\_column\_letter

import xlrd

import xlwt

from xlwt import Workbook

import xlsxwriter

import os

from datetime import datetime

import re

import psycopg2

import warnings

from flask\_sqlalchemy import SQLAlchemy

warnings.filterwarnings('ignore')

from datetime import datetime

#import timedelta

from datetime import timedelta

#import local functions and models

import configparser

config = configparser.ConfigParser()

config.read('config/settings.ini')

import functions

import functions2

import model\_vtz

path = 'doc/VTZ\_pipes.xlsx'

df=pd.read\_excel(path, sheet\_name='model')

list\_marki=pd.read\_excel(path, sheet\_name='marki')

list\_kalibr=pd.read\_excel(path, sheet\_name='kalibr')

data\_prop=pd.read\_excel('doc/data\_properties\_VTZ.xlsx',sheet\_name='скорости\_все')

df\_for\_model=pd.read\_excel(path, sheet\_name='match\_with\_model')

except\_model=pd.read\_excel('doc/except.xlsx')

def ff(x):

return list(x)[0]

list\_marki\_1 = list\_marki['marki'].unique()

list\_kalibr\_1 = list\_kalibr['kalibr'].unique()

#TABLE FOR SELECT ID

def choose\_table\_create(df\_sql):

if len(df\_sql.columns) >1:

tb = df\_sql[['scenarioid', 'datetime', 'vtz\_report1\_1', 'vtz\_report1\_12',

'vtz\_report1\_3', 'vtz\_report2\_1', 'vtz\_report2\_3']]

try:

tb['datetime'] = tb['datetime'].astype('datetime64[ns]').dt.round('S').dt.strftime('%Y-%m-%d %H:%M:%S')

tb['datetime'] = tb['datetime'].astype('string')

except:

tb['datetime'] = tb['datetime']

else:

tb = df\_sql

tb['Дата создания'] = None

tb['Калибр, мм'] = None

tb['Код отправки'] = None

tb['Марка стали'] = None

tb['Диаметр трубы, мм'] = None

tb['Толщина стенки, мм'] = None

return tb

head = functions2.sql\_data\_LIMIT500(order\_id = 'datetime', limit = 5).loc[0].T #default table - 'data\_report'

head = pd.DataFrame(head)

head = head.reset\_index()

head.rename(columns = {'index':'name', 0:'vals'}, inplace = True)

###первая таблица

def display\_tb(df, df2):

df1 = df.copy()

if df2.empty == True:

tb = df[df['num'].isin(range(9))]

tb = tb.sort\_values(by=['num'])

line = pd.DataFrame({"name\_properties\_2": 'Заготовка', "Значение": None}, index=[5])

tb = pd.concat([tb.iloc[:5], line, tb.iloc[5:]]).reset\_index(drop=True)

tb = tb.drop(4)

tb['Значение'] = None

tb2 = df[df['num'].isin(range(9,19))]

tb2 = tb2.sort\_values(by=['num'])

tb2['Значение'] = None

tb3 = df[df['num'].isin(range(19,41))]

tb3 = tb3.sort\_values(by=['num'])

tb3['Значение'] = None

tb4 = df[df['num'].isin(range(41,68))]

tb4 = tb4.sort\_values(by=['num'])

tb4['Значение'] = None

else:

df2 = df2.T

df2 = df2.reset\_index()

df2.rename(columns = {'index': 'Report', 0: 'Значение'}, inplace = True)

df2['Report'] = df2.Report.replace({'vtz':'VTZ'}, regex=True)

df3 = df1.merge(df2, left\_on='Report', right\_on='Report', how = 'left')

val = df3['Значение'][4].split(sep = '/')

df3['Значение'][0] = val[1]

df3['Значение'][1] = val[0]

tb = df3[df3['num'].isin(range(9))]

tb = tb.sort\_values(by=['num'])

line = pd.DataFrame({"name\_properties\_2": 'Заготовка', "Значение": None}, index=[5])

tb = pd.concat([tb.iloc[:5], line, tb.iloc[5:]]).reset\_index(drop=True)

tb2 = df3[df3['num'].isin(range(9,19))]

tb2 = tb2.sort\_values(by=['num'])

tb3 = df3[df3['num'].isin(range(19,41))]

tb3 = tb3.sort\_values(by=['num'])

tb4 = df3[df3['num'].isin(range(41,68))]

tb4 = tb4.sort\_values(by=['num'])

tb = tb[['name\_properties\_2', 'Значение']]

tb = tb.rename(columns = {'name\_properties\_2': 'Параметр', 'Значение': 'Значение'}, inplace = False)

tb = tb.reset\_index(drop=True)

tb2 = tb2[['name\_properties\_2', 'Значение']]

tb2 = tb2.rename(columns = {'name\_properties\_2': 'Параметр', 'Значение': 'Значение'}, inplace = False)

tb2 = tb2.reset\_index(drop=True)

tb3 = tb3[['name\_properties\_2', 'Значение']]

tb3 = tb3.rename(columns = {'name\_properties\_2': 'Параметр', 'Значение': 'Значение'}, inplace = False)

tb3 = tb3.reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #1', "Значение": None}, index=[0])

tb3 = pd.concat([line, tb3]).reset\_index(drop=True)

tb3 = tb3.reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #2', "Значение": None}, index=[4])

tb3 = pd.concat([tb3.iloc[:4], line, tb3.iloc[4:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #3', "Значение": None}, index=[8])

tb3 = pd.concat([tb3.iloc[:8], line, tb3.iloc[8:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #4', "Значение": None}, index=[12])

tb3 = pd.concat([tb3.iloc[:12], line, tb3.iloc[12:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #5', "Значение": None}, index=[16])

tb3 = pd.concat([tb3.iloc[:16], line, tb3.iloc[16:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #6', "Значение": None}, index=[20])

tb3 = pd.concat([tb3.iloc[:20], line, tb3.iloc[20:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #7', "Значение": None}, index=[24])

tb3 = pd.concat([tb3.iloc[:24], line, tb3.iloc[24:]]).reset\_index(drop=True)

tb4 = tb4[['name\_properties\_2', 'Значение']]

tb4 = tb4.rename(columns = {'name\_properties\_2': 'Параметр', 'Значение': 'Значение'}, inplace = False)

tb4 = tb4.reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #1', "Значение": None}, index=[0])

tb4 = pd.concat([line, tb4]).reset\_index(drop=True)

tb4 = tb4.reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #2', "Значение": None}, index=[4])

tb4 = pd.concat([tb4.iloc[:4], line, tb4.iloc[4:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #3', "Значение": None}, index=[8])

tb4 = pd.concat([tb4.iloc[:8], line, tb4.iloc[8:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #4', "Значение": None}, index=[12])

tb4 = pd.concat([tb4.iloc[:12], line, tb4.iloc[12:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #5', "Значение": None}, index=[16])

tb4 = pd.concat([tb4.iloc[:16], line, tb4.iloc[16:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #6', "Значение": None}, index=[20])

tb4 = pd.concat([tb4.iloc[:20], line, tb4.iloc[20:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #7', "Значение": None}, index=[24])

tb4 = pd.concat([tb4.iloc[:24], line, tb4.iloc[24:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #8', "Значение": None}, index=[28])

tb4 = pd.concat([tb4.iloc[:28], line, tb4.iloc[28:]]).reset\_index(drop=True)

line = pd.DataFrame({"Параметр": 'Клеть #9', "Значение": None}, index=[32])

tb4 = pd.concat([tb4.iloc[:32], line, tb4.iloc[32:]]).reset\_index(drop=True)

return tb, tb2, tb3, tb4

#функция для переноса данных в эксель

def template(replacementTextKeyPairs, card\_name):

book = opxl.load\_workbook('./doc/' + card\_name)

ws = book.worksheets[0]

number\_of\_rows = ws.max\_row

number\_of\_cols = ws.max\_column

for i in range(number\_of\_cols):

for k in range(number\_of\_rows):

cellValue = str(ws[get\_column\_letter(i+1)+str(k+1)].value)

for key in replacementTextKeyPairs.keys():

if str(cellValue) == key:

newCellValue = replacementTextKeyPairs.get(key)

ws[get\_column\_letter(i+1)+str(k+1)] = str(newCellValue)

book.save(card\_name)

return book.save(card\_name)

###CREATE FIGURE \_APP MODEL

def graph\_1(x\_data, y\_data):

#config={'responsive': True}

fig=make\_subplots(rows=1,cols=1)

fig.update\_layout(barmode='relative',

#title\_text='график трубы',

plot\_bgcolor = '#ecf0f1',

#plot\_bgcolor = '#B0C4DE',

paper\_bgcolor = '#ecf0f1',

yaxis\_title ='Cкорость трубы в клети, мм/сек',

xaxis\_title ='Номер клети',

#height='400',

autosize = True,

margin=dict(

l=20,

r=35,

b=5,

t=40,

pad=4),

)

fig.add\_scatter(x=x\_data, y=y\_data, name="исторические данные", mode='lines',

line=dict(color='#191970', width=3),

row=1,col=1)

return fig

def graph\_2(x\_data, y\_data, z\_data):

fig=make\_subplots(rows=1,cols=1)

fig.update\_layout(barmode='relative',

#title\_text='график трубы',

plot\_bgcolor = '#ecf0f1',

#plot\_bgcolor = '#B0C4DE',

paper\_bgcolor = '#ecf0f1',

yaxis\_title ='Cкорость трубы в клети, мм/сек',

xaxis\_title ='Номер клети',

#height='400',

showlegend = True,

legend=dict(

orientation="h",

yanchor="bottom",

y=1.1,

xanchor="left",

x=0.25

),

autosize = True,

margin=dict(

l=20,

r=35,

b=5,

t=40,

pad=4),

)

fig.add\_scatter(x=x\_data, y=y\_data, name='Данные расчетные', mode='lines',

line=dict(color='#191970', width=3),

row=1,col=1)

fig.add\_scatter(x=x\_data, y=z\_data, name='Данные аналоги', mode='lines',

line=dict(color='red', width=3),

row=1,col=1)

return fig

data\_pipes=pd.read\_excel('doc/excel\_pipes.xlsx')

name\_list\_pipes=[item.replace('\xa0',' ') for item in list(data\_pipes.columns)]

data\_pipes.columns=name\_list\_pipes

for i in data\_pipes.columns:

try:data\_pipes[i]=data\_pipes[i].astype('float')

except: ...

data\_pipes['id']=data\_pipes['Диаметр трубы'].apply(lambda x: format(x, '.2f'))+'\_'+\

data\_pipes['Стенка трубы'].apply(lambda x: format(x, '.2f'))+'\_'+\

data\_pipes['Диаметр оправки (мм)\_MPM'].apply(lambda x: format(x, '.2f'))+'\_'+\

data\_pipes['Калибр \_ELM'].apply(lambda x: format(x, '.2f'))+'\_'+\

data\_pipes['марка стали'].astype('str')

data\_pipes=data\_pipes[[data\_pipes.columns[-1]]+list(data\_pipes.columns[:-1])]

list\_name\_db=list(functions.read\_sql\_ff('dash\_pipes').columns)

data\_name=pd.DataFrame({'name\_dash':list(data\_pipes.columns),'name\_bd':list\_name\_db,'type':list(data\_pipes.dtypes)})

data\_name.loc[data\_name['type']=='float64','type']='numeric'

data\_name.loc[data\_name['type']=='object','type']='text'

cols0=[{"name": data\_name['name\_dash'][i], "id": data\_name['name\_bd'][i], 'type': data\_name['type'][i]} for i in range(len(data\_name)) ]

def get\_nearest\_value(iterable, value):

return min(iterable, key=lambda x: abs(x - value))

Kalibr=0

diam\_fact\_mpm=0

def ff\_speed(zazor,oborot,Kalibr=Kalibr,diam\_fact\_mpm=diam\_fact\_mpm,data\_prop=data\_prop):

data\_speed=data\_prop.loc[data\_prop['Калибр']==get\_nearest\_value(data\_prop['Калибр'], Kalibr)]

data\_speed['Диаметр бочки валка, мм']=diam\_fact\_mpm

data\_speed['зазор, мм']=zazor

data\_speed['обороты двигателя, об/мин']=oborot

data\_speed['Скорость валков, об/мин']=data\_speed['обороты двигателя, об/мин']/data\_speed['передаточное число редуктора']

data\_speed['диаметр калибра валка МРМ, мм']=2\*data\_speed['высота калибра, мм']+data\_speed['зазор, мм']

data\_speed['Катающий диаметр, мм']=data\_speed['Диаметр бочки валка, мм']+data\_speed['зазор, мм']-data\_speed['Коэффициент формы калибра']\*data\_speed['диаметр калибра валка МРМ, мм']

data\_speed['скорость трубы в клети, мм/сек']=3.14\*data\_speed['Катающий диаметр, мм']\*data\_speed['Скорость валков, об/мин']/60

fig=make\_subplots(rows=1,cols=1)

fig = px.line(data\_speed, x="№ клети", y="скорость трубы в клети, мм/сек")

fig.update\_layout(barmode='relative',

autosize = True,

yaxis\_title ='Cкорость трубы в клети, мм/сек',

xaxis\_title ='Номер клети',

margin=dict(

l=10,

r=20,

b=100,

t=40,

pad=4),)

# fig.show()

# return list(data\_speed['скорость трубы в клети, мм/сек'])

return fig

def table\_1(data,row,selected\_rows=[],data\_name=data\_name):

if (row!=None) and (row!=[]):

data2=data.loc[[row[0]]]

############### table

table\_1=dash\_table.DataTable(

id='out-table',

columns=cols0,

data=data.to\_dict('records'),

page\_size=10,

style\_cell\_conditional=[{'if': {'column\_id': c},'display': 'none'} for c in [list(data\_name['name\_bd'])[0]]+list(data\_name['name\_bd'])[6:]],

# editable=editable,

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

'overflow': 'hidden',

'textAlign': 'center',

},

row\_deletable=False,

filter\_action="native",

sort\_action="native",

row\_selectable="single",

sort\_mode="single",

page\_action='native',

selected\_rows=selected\_rows,

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 22px'}],

style\_header={

'textAlign': 'center',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",},

page\_current= 0)

############### table2

data\_mpm\_esm=pd.DataFrame({'стан':['МПМ','МПМ','МПМ','МПМ','МПМ','МПМ','МПМ','ИКС','ИКС','ИКС','ИКС','ИКС','ИКС','ИКС','ИКС','ИКС']})

data\_mpm\_esm['Номер клети']=[1,2,3,4,5,6,7,1,2,3,4,5,6,7,8,9]

data\_mpm\_esm['Диаметр Валков']=list(data2[['diametrvalkovmmklet1mpm','diametrvalkovmmklet2mpm',\

'diametrvalkovmmklet3mpm','diametrvalkovmmklet4mpm',\

'diametrvalkovmmklet5mpm','diametrvalkovmmklet6mpm','diametrvalkovmmklet7mpm',\

'diametrvalkovmmklet1esm','diametrvalkovmmklet2esm',\

'diametrvalkovmmklet3esm','diametrvalkovmmklet4esm','diametrvalkovmmklet5esm',\

'diametrvalkovmmklet6esm','diametrvalkovmmklet7esm',\

'diametrvalkovmmklet8esm','diametrvalkovmmklet9esm']].mean())

data\_mpm\_esm['Обороты']=list(data2[['skorostvalkovklet1obminmpm','skorostvalkovklet2obminmpm',\

'skorostvalkovklet3obminmpm',\

'skorostvalkovklet4obminmpm','skorostvalkovklet5obminmpm','skorostvalkovklet6obminmpm',\

'skorostvalkovklet7obminmpm','skorostkletiobminklet1esm',\

'skorostkletiobminklet2esm','skorostkletiobminklet3esm','skorostkletiobminklet4esm',\

'skorostkletiobminklet5esm','skorostkletiobminklet6esm',\

'skorostkletiobminklet7esm','skorostkletiobminklet8esm','skorostkletiobminklet9esm']].mean())

data\_mpm\_esm['Зазоры']=list(data2[['zazorvalkovmmklet1mpm', 'zazorvalkovmmklet2mpm',\

'zazorvalkovmmklet3mpm','zazorvalkovmmklet4mpm',\

'zazorvalkovmmklet5mpm','zazorvalkovmmklet6mpm', 'zazorvalkovmmklet7mpm',\

'zazorvalkovmmklet1esm', 'zazorvalkovmmklet2esm','zazorvalkovmmklet3esm',\

'zazorvalkovmmklet4esm', 'zazorvalkovmmklet5esm', 'zazorvalkovmmklet6esm', 'zazorvalkovmmklet7esm',\

'zazorvalkovmmklet8esm', 'zazorvalkovmmklet9esm']].mean())

############### plot\_1

zazor=data2[['zazorvalkovmmklet1mpm','zazorvalkovmmklet2mpm','zazorvalkovmmklet3mpm','zazorvalkovmmklet4mpm',\

'zazorvalkovmmklet5mpm', 'zazorvalkovmmklet6mpm','zazorvalkovmmklet7mpm']]

zazor=list(zazor.mean())

oborot=data2[['skorostvalkovklet1obminmpm','skorostvalkovklet2obminmpm','skorostvalkovklet3obminmpm',\

'skorostvalkovklet4obminmpm', 'skorostvalkovklet5obminmpm','skorostvalkovklet6obminmpm','skorostvalkovklet7obminmpm']]

oborot=list(oborot.mean())

Kalibr=list(data2['kalibrelm'])[0]

diam\_fact\_mpm=data2[['diametrvalkovmmklet1mpm','diametrvalkovmmklet2mpm','diametrvalkovmmklet3mpm',\

'diametrvalkovmmklet4mpm',\

'diametrvalkovmmklet5mpm','diametrvalkovmmklet6mpm','diametrvalkovmmklet7mpm']]

diam\_fact\_mpm=list(diam\_fact\_mpm.mean())

list\_a = ["стан", 'Номер клети']

cols\_tb2 = [

{"name": i, "id": i, "editable": False,} for i in data\_mpm\_esm.columns if i in list\_a]

cols\_tb2\_2 = [

{"name": i, "id": i, 'type': 'numeric', "editable": True,

} for i in data\_mpm\_esm.columns if i not in list\_a]

cols\_tb2.extend(cols\_tb2\_2)

table\_2=dash\_table.DataTable(

id='out-table2',

#columns=[{"name": i, "id": i} for i in data\_mpm\_esm.columns],

columns=cols\_tb2,

data=data\_mpm\_esm.to\_dict('records'),

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 22px'}],

style\_header={

'textAlign': 'center',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",},

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

'overflow': 'hidden',

'textAlign': 'center',

},

style\_cell\_conditional=[{'if': {'row\_index': 'odd'},

'backgroundColor': "#F5F5F5",}],

editable=True)

plot\_1=dcc.Graph(figure=ff\_speed(zazor=zazor,

oborot=oborot,

Kalibr=Kalibr,

diam\_fact\_mpm=diam\_fact\_mpm,

data\_prop=data\_prop))

return table\_1,table\_2,plot\_1

else:

table\_1=dash\_table.DataTable(

id='out-table',

columns=cols0,

data=data.to\_dict('records'),

page\_size=10,

style\_cell\_conditional=[{'if': {'column\_id': c},'display': 'none'} for c in [list(data\_name['name\_bd'])[0]]+list(data\_name['name\_bd'])[6:]],

# editable=editable,

row\_deletable=False,

filter\_action="native",

sort\_action="native",

row\_selectable="single",

sort\_mode="single",

page\_action='native',

selected\_rows=selected\_rows,

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 22px'}],

style\_header={

'textAlign': 'center',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",},

page\_current= 0)

return table\_1,no\_update,no\_update

table\_2=dash\_table.DataTable(id='out-table2',data=pd.DataFrame().to\_dict('records'),

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 22px'}],

style\_header={

'textAlign': 'center',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",},)

fig=make\_subplots(rows=1,cols=1)

fig.update\_layout(barmode='relative', title\_text='график трубы')

fig.add\_scatter(x=[1,2,3,4,5,6,7], y=[830,1300,1600,1800,1870,1900,1860],mode='lines',name="исторические данные",row=1,col=1)

plot\_1=dcc.Graph(figure=fig)

def df\_to\_csv\_2(row,dataset,dataset2):

pg = pd.DataFrame(dataset)

pg2 = pd.DataFrame(dataset2)

pg2['Диаметр Валков']=pg2['Диаметр Валков'].astype('float')

pg2['Обороты']=pg2['Обороты'].astype('float')

pg2['Зазоры']=pg2['Зазоры'].astype('float')

pg.loc[row,['diametrvalkovmmklet1mpm','diametrvalkovmmklet2mpm','diametrvalkovmmklet3mpm','diametrvalkovmmklet4mpm',\

'diametrvalkovmmklet5mpm','diametrvalkovmmklet6mpm','diametrvalkovmmklet7mpm','diametrvalkovmmklet1esm','diametrvalkovmmklet2esm',\

'diametrvalkovmmklet3esm','diametrvalkovmmklet4esm','diametrvalkovmmklet5esm','diametrvalkovmmklet6esm','diametrvalkovmmklet7esm',\

'diametrvalkovmmklet8esm','diametrvalkovmmklet9esm']]=list(pg2['Диаметр Валков'])

pg.loc[row,['skorostvalkovklet1obminmpm','skorostvalkovklet2obminmpm','skorostvalkovklet3obminmpm',\

'skorostvalkovklet4obminmpm','skorostvalkovklet5obminmpm','skorostvalkovklet6obminmpm','skorostvalkovklet7obminmpm','skorostkletiobminklet1esm',\

'skorostkletiobminklet2esm','skorostkletiobminklet3esm','skorostkletiobminklet4esm','skorostkletiobminklet5esm','skorostkletiobminklet6esm',\

'skorostkletiobminklet7esm','skorostkletiobminklet8esm','skorostkletiobminklet9esm']]=list(pg2['Обороты'])

pg.loc[row,['zazorvalkovmmklet1mpm', 'zazorvalkovmmklet2mpm', 'zazorvalkovmmklet3mpm','zazorvalkovmmklet4mpm',\

'zazorvalkovmmklet5mpm','zazorvalkovmmklet6mpm', 'zazorvalkovmmklet7mpm','zazorvalkovmmklet1esm', 'zazorvalkovmmklet2esm','zazorvalkovmmklet3esm',\

'zazorvalkovmmklet4esm', 'zazorvalkovmmklet5esm', 'zazorvalkovmmklet6esm', 'zazorvalkovmmklet7esm', 'zazorvalkovmmklet8esm',\

'zazorvalkovmmklet9esm']]=list(pg2['Зазоры'])

pg['id']=pg['diametrtruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['stenkatruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['diametropravkimmmpm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['kalibrelm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['markastali'].astype('str')

pg=pg.replace('', 0)

for i in pg.columns:

if i=='markastali' or i=='id':

pg[i]=pg[i].astype('object')

else:

pg[i]=pg[i].astype('float')

functions.update\_sql\_ff(pg,'dash\_pipes')

return html.Plaintext("данные сохранены "+str(datetime.now() + timedelta(hours=3)),style={'color': 'red', 'font-weight': 'bold', 'font-size': 'large'})

diam\_fact\_mpm= [916,894,912,817,818,826,827]

diam\_fact\_mpm\_new= [1016,994,1012,917,918,926,927]

def return\_speed(zazor,oborot,Kalibr=Kalibr,diam\_fact\_mpm=diam\_fact\_mpm,data\_prop=data\_prop,diam\_fact\_mpm\_new=diam\_fact\_mpm\_new):

data\_speed=data\_prop.loc[data\_prop['Калибр']==get\_nearest\_value(data\_prop['Калибр'], Kalibr)]

data\_speed['Диаметр бочки валка, мм']=diam\_fact\_mpm

data\_speed['зазор, мм']=zazor

data\_speed['обороты двигателя, об/мин']=oborot

data\_speed['Скорость валков, об/мин']=data\_speed['обороты двигателя, об/мин']/data\_speed['передаточное число редуктора']

data\_speed['диаметр калибра валка МРМ, мм']=2\*data\_speed['высота калибра, мм']+data\_speed['зазор, мм']

data\_speed['Катающий диаметр, мм']=data\_speed['Диаметр бочки валка, мм']+data\_speed['зазор, мм']-data\_speed['Коэффициент формы калибра']\*data\_speed['диаметр калибра валка МРМ, мм']

data\_speed['скорость трубы в клети, мм/сек']=3.14\*data\_speed['Катающий диаметр, мм']\*data\_speed['Скорость валков, об/мин']/60

speed=list(data\_speed['скорость трубы в клети, мм/сек'])

###

np\_array = np.array(speed)

np\_round\_ = np.around(np\_array, 1)

speed = list(np\_round\_)

data\_speed=data\_prop.loc[data\_prop['Калибр']==get\_nearest\_value(data\_prop['Калибр'], Kalibr)]

data\_speed['Диаметр бочки валка, мм']=diam\_fact\_mpm\_new

data\_speed['зазор, мм']=zazor

data\_speed['диаметр калибра валка МРМ, мм']=2\*data\_speed['высота калибра, мм']+data\_speed['зазор, мм']

data\_speed['Катающий диаметр, мм']=data\_speed['Диаметр бочки валка, мм']+data\_speed['зазор, мм']\

-data\_speed['Коэффициент формы калибра']\*data\_speed['диаметр калибра валка МРМ, мм']

MPM\_oborot=list(pd.Series(speed)\*60/3.14/data\_speed['Катающий диаметр, мм'].reset\_index(drop=True)\*data\_speed['передаточное число редуктора'].reset\_index(drop=True))

np\_array = np.array(MPM\_oborot)

np\_round\_ = np.around(np\_array, 0)

MPM\_oborot = list(np\_round\_)

diametr\_cat = list(data\_speed['Катающий диаметр, мм'])

np\_array = np.array(diametr\_cat)

np\_round\_ = np.around(np\_array, 1)

diametr\_cat = list(np\_round\_)

return speed,MPM\_oborot,diametr\_cat #скорости, обороты

def create\_analog\_tabl(df):

table = dash\_table.DataTable(

id='table\_analog',

columns= [

#{'name': 'Время создания', 'id': 'mpmin', "editable": False},

{'name': 'Время создания', 'id': 'Data', "editable": False},

{'name': 'Марки стали', 'id': 'markastali', "editable": False},

{'name': 'Источник данных', 'id': 'Источник данных', "editable": False},

{'name': 'ID', 'id': 'ID', "editable": False},],

data = df.to\_dict('record'),

row\_selectable="single",

page\_size=10,

selected\_rows=[],

filter\_action='native',

page\_action='native',

style\_as\_list\_view=True,

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

'font-size': '14px'

},

style\_data\_conditional=[

{

'if': {'row\_index': 'odd'},

'backgroundColor': "#c7e5f3",

}],

style\_cell\_conditional=[

{'if': {'column\_id': 'ID'},'display': 'none'},

{'if': {'column\_id': 'Data'},'textAlign': 'center'},

{'if': {'column\_id': 'markastali'},'textAlign': 'center'},

{'if': {'column\_id': 'Источник данных'},'textAlign': 'center'},

],

style\_header={

#"backgroundColor": "#353A40", #292D32

'textAlign': 'center',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",

},

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 18px'}],

# fixed\_rows={'headers': True},

)

return table

#df для таблицы с зазорами и оборотами. Передает стрку из БД (текущии) + зазоры + обороты аналогов

def tables\_zazory(df, df2, skor, zazor):

df1 = df.copy()

df2 = df2.T

df2 = df2.reset\_index()

df2.rename(columns = {'index': 'Report', 0: 'Значение'}, inplace = True)

df2['Report'] = df2.Report.replace({'vtz':'VTZ'}, regex=True)

df3 = df1.merge(df2, left\_on='Report', right\_on='Report', how = 'left')

tb = df3[df3['num'].isin(range(19,68))]

tb = tb.sort\_values(by=['num'])

tb = tb[['name\_properties\_2', 'Значение']]

tb = tb.rename(columns = {'name\_properties\_2': 'Параметр', 'Значение': 'Значение'}, inplace = False)

tb = tb.reset\_index(drop=True)

df\_zazor = pd.DataFrame()

df\_zazor['Стан'] = ['МПМ']\*7 + ['ИКС']\*9

df\_zazor['Номер клети'] = list(range(1, 8)) + list(range(1,10))

df\_zazor['Диаметр Валков'] = tb.loc[tb['Параметр'] =='Актуал. диаметр валка, мм', 'Значение'].reset\_index(drop=True)

df\_zazor['Обороты расчетные'] = tb.loc[tb['Параметр'] =='Скорость двигателей, об/мин', 'Значение'].reset\_index(drop=True)

df\_zazor['Зазоры расчетные'] = tb.loc[tb['Параметр'] =='Зазор валков, мм', 'Значение'].reset\_index(drop=True)

df\_zazor['Обороты аналоги'] = skor

df\_zazor['Зазоры аналоги'] = zazor

df\_zazor['Диаметр Валков'] = (pd.to\_numeric(df\_zazor['Диаметр Валков'], downcast="float", errors='coerce').fillna(0)).astype(float).round(0)

df\_zazor['Обороты расчетные'] = (pd.to\_numeric(df\_zazor['Обороты расчетные'], downcast="float", errors='coerce').fillna(0)).astype(float).round(1)

df\_zazor['Обороты аналоги'] = (pd.to\_numeric(df\_zazor['Обороты аналоги'], downcast="float", errors='coerce').fillna(0)).astype(float).round(1)

df\_zazor['Зазоры расчетные'] = (pd.to\_numeric(df\_zazor['Зазоры расчетные'], downcast="float", errors='coerce').fillna(0)).astype(float).round(1)

df\_zazor['Зазоры аналоги'] = (pd.to\_numeric(df\_zazor['Зазоры аналоги'], downcast="float", errors='coerce').fillna(0)).astype(float).round(1)

df\_zazor.replace([np.inf, -np.inf, np.nan], 0, inplace=True)

df\_zazor['Зазоры расчетные'] = df\_zazor.apply(lambda row: 0 if row['Диаметр Валков'] == 0 else row['Зазоры расчетные'], axis=1)

df\_zazor['Обороты расчетные'] = df\_zazor.apply(lambda row: 0 if row['Диаметр Валков'] == 0 else row['Обороты расчетные'], axis=1)

df\_zazor['Обороты аналоги'] = df\_zazor.apply(lambda row: 0 if row['Диаметр Валков'] == 0 else row['Обороты аналоги'], axis=1)

df\_zazor['Зазоры аналоги'] = df\_zazor.apply(lambda row: 0 if row['Диаметр Валков'] == 0 else row['Зазоры аналоги'], axis=1)

df\_zazor['Разница зазоры'] = (df\_zazor['Зазоры расчетные'] - df\_zazor['Зазоры аналоги']).astype(float).round(1)

df\_zazor['Разница обороты,%'] = ((df\_zazor['Обороты расчетные'] - df\_zazor['Обороты аналоги'])/df\_zazor['Обороты расчетные'] \*100).astype(float).round(1)

df\_zazor.replace([np.inf, -np.inf, np.nan], 0, inplace=True)

return df\_zazor

def table\_realpipes(real\_pipes, row):

try:

real\_pipes = real\_pipes.rename(columns={'timestamp': 'Дата обкатки',

'diametropravkimmmpm': 'Диаметр оправки (мм)\_МРМ',

'diametrtruby': 'Диаметр трубы',

'stenkatruby': 'Стенка трубы',

'markastali': 'Марка стали',

'kalibrelm': 'Калибр\_ELM'})

cols = list(real\_pipes.columns)

cols[3], cols[4] = cols[4], cols[3]

real\_pipes = real\_pipes[cols]

except:

real\_pipes = pd.DataFrame(columns=['Дата обкатки', 'Диаметр оправки (мм)\_МРМ',

'Диаметр трубы', 'Стенка трубы', 'Марка стали', 'Калибр\_ELM'])

table\_1= dash\_table.DataTable(

id='real\_pipes',

columns= [{"name": i, "id": i}

for i in real\_pipes.columns],

data=real\_pipes.to\_dict('records'),

page\_size=10,

style\_cell\_conditional=[{'if': {'column\_id': c},'display': 'none'}

for c in list(real\_pipes.columns)[6:]],

# editable=editable,

row\_deletable=False,

filter\_action="native",

sort\_action="native",

row\_selectable="single",

sort\_mode="single",

page\_action='native',

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 22px'}],

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

'overflow': 'hidden',

'textAlign': 'center',

# 'font-size': '12px'

},

style\_header={

'textAlign': 'center',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",},

page\_current= 0),

############### table2

if len(real\_pipes) >0:

data2=real\_pipes.loc[row]

Kalibr=data2['Калибр\_ELM']

data\_mpm\_esm=pd.DataFrame({'стан':['МПМ','МПМ','МПМ','МПМ','МПМ','МПМ','МПМ','ИКС','ИКС','ИКС','ИКС','ИКС','ИКС','ИКС','ИКС','ИКС']})

data\_mpm\_esm['Номер клети']=[1,2,3,4,5,6,7,1,2,3,4,5,6,7,8,9]

data\_mpm\_esm['Диаметр Валков']=list(data2[['diametrvalkovmmklet1mpm','diametrvalkovmmklet2mpm','diametrvalkovmmklet3mpm','diametrvalkovmmklet4mpm',\

'diametrvalkovmmklet5mpm','diametrvalkovmmklet6mpm','diametrvalkovmmklet7mpm','diametrvalkovmmklet1esm','diametrvalkovmmklet2esm',\

'diametrvalkovmmklet3esm','diametrvalkovmmklet4esm','diametrvalkovmmklet5esm','diametrvalkovmmklet6esm','diametrvalkovmmklet7esm',\

'diametrvalkovmmklet8esm','diametrvalkovmmklet9esm']])

data\_mpm\_esm['Обороты']=list(data2[['skorostvalkovklet1obminmpm','skorostvalkovklet2obminmpm','skorostvalkovklet3obminmpm',\

'skorostvalkovklet4obminmpm','skorostvalkovklet5obminmpm','skorostvalkovklet6obminmpm','skorostvalkovklet7obminmpm','skorostkletiobminklet1esm',\

'skorostkletiobminklet2esm','skorostkletiobminklet3esm','skorostkletiobminklet4esm','skorostkletiobminklet5esm','skorostkletiobminklet6esm',\

'skorostkletiobminklet7esm','skorostkletiobminklet8esm','skorostkletiobminklet9esm']])

data\_mpm\_esm['Зазоры']=list(data2[['zazorvalkovmmklet1mpm', 'zazorvalkovmmklet2mpm', 'zazorvalkovmmklet3mpm','zazorvalkovmmklet4mpm',\

'zazorvalkovmmklet5mpm','zazorvalkovmmklet6mpm', 'zazorvalkovmmklet7mpm','zazorvalkovmmklet1esm', 'zazorvalkovmmklet2esm','zazorvalkovmmklet3esm',\

'zazorvalkovmmklet4esm', 'zazorvalkovmmklet5esm', 'zazorvalkovmmklet6esm', 'zazorvalkovmmklet7esm', 'zazorvalkovmmklet8esm',\

'zazorvalkovmmklet9esm']])

# ############### plot\_1

zazor=data2[['zazorvalkovmmklet1mpm','zazorvalkovmmklet2mpm','zazorvalkovmmklet3mpm','zazorvalkovmmklet4mpm','zazorvalkovmmklet5mpm',\

'zazorvalkovmmklet6mpm','zazorvalkovmmklet7mpm']]

oborot=data2[['skorostvalkovklet1obminmpm','skorostvalkovklet2obminmpm','skorostvalkovklet3obminmpm','skorostvalkovklet4obminmpm',\

'skorostvalkovklet5obminmpm','skorostvalkovklet6obminmpm','skorostvalkovklet7obminmpm']]

diam\_fact\_mpm=data2[['diametrvalkovmmklet1mpm','diametrvalkovmmklet2mpm','diametrvalkovmmklet3mpm','diametrvalkovmmklet4mpm',\

'diametrvalkovmmklet5mpm','diametrvalkovmmklet6mpm','diametrvalkovmmklet7mpm']]

plot\_1=dcc.Graph(figure=ff\_speed(zazor=list(zazor.astype(float)),

oborot=list(oborot.astype(float)),

Kalibr=float(Kalibr),

diam\_fact\_mpm=list(diam\_fact\_mpm.astype(float)),

data\_prop=data\_prop))

else:

plot\_1= None

data\_mpm\_esm = pd.DataFrame()

list\_a = ["стан", 'Номер клети']

cols\_tb2 = [{"name": i, "id": i, "editable": False,} for i in data\_mpm\_esm.columns if i in list\_a]

cols\_tb2\_2 = [{"name": i, "id": i, 'type': 'numeric', "editable": True,} for i in data\_mpm\_esm.columns if i not in list\_a]

cols\_tb2.extend(cols\_tb2\_2)

table\_2=dash\_table.DataTable(

id='real\_pipes2',

#columns=[{"name": i, "id": i} for i in data\_mpm\_esm.columns],

columns=cols\_tb2,

data=data\_mpm\_esm.to\_dict('records'),

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 22px'}],

style\_header={

'textAlign': 'center',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",},

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

'overflow': 'hidden',

'textAlign': 'center',

# 'font-size': '12px'

},

style\_cell\_conditional=[{'if': {'row\_index': 'odd'},

'backgroundColor': "#F5F5F5",}],

editable=True)

return table\_1,table\_2,plot\_1

def currents\_graph(currents):

#change data types to float

for col in range(1,(currents.shape[1])):

name = currents.columns[col]

try:

currents[name] = currents[name].astype("float")

except:

currents[name] = currents[name].astype("object")

try:

fig=make\_subplots(rows=1,cols=1,specs=[[{"secondary\_y": True}]])

for i in range(1,8):

fig.add\_scatter(x=currents['timestamp'], y=currents['tas\_mpm\_curr\_std'+str(i)], name='ток'+str(i), mode='lines',row=1,col=1,secondary\_y=False)

fig.update\_layout(barmode='relative',

title={

'text': "Токи mpm",

'y':0.99,

'x':0.5,

'xanchor': 'center',

'yanchor': 'top'},

font=dict(

# family="Courier New, monospace",

# size=18,

color="grey"),

legend=dict(

# orientation="h",

# yanchor="bottom",

# y=1.01,

# xanchor="left",

# x=0.0,

font=dict(size=10,),

),

autosize = True,

margin=dict(

l=5,

r=5,

b=5,

t=40,

pad=4),

)

toki\_mpm = fig

except:

toki\_mpm = None

try:

fig=make\_subplots(rows=1,cols=1,specs=[[{"secondary\_y": True}]])

for i in range(1,10):

fig.add\_scatter(x=currents['timestamp'], y=currents['tas\_esm\_curr\_std'+str(i)], name='ток'+str(i),mode='lines',row=1,col=1,secondary\_y=False)

fig.update\_layout(barmode='relative',

title={

'text': "Токи esm",

'y':0.99,

'x':0.5,

'xanchor': 'center',

'yanchor': 'top'},

font=dict(

# family="Courier New, monospace",

# size=18,

color="grey"),

legend=dict(

# orientation="h",

# yanchor="bottom",

# y=1.01,

# xanchor="left",

# x=0.0,

font=dict(size=10,),

),

autosize = True,

margin=dict(

l=5,

r=5,

b=5,

t=40,

pad=4),

)

toki\_esm = fig

except:

toki\_esm = None

if len(currents) > 0:

if ff(currents.loc[:,'tas\_mpm\_curr\_std1':'tas\_mpm\_curr\_std7'].T.isnull().sum()) == 7:

toki\_mpm = None

if ff(currents.loc[:,'tas\_esm\_curr\_std1':'tas\_esm\_curr\_std9'].T.isnull().sum()) == 9:

toki\_esm = None

elif len(currents) == 0:

toki\_mpm = None

toki\_esm = None

return toki\_mpm, toki\_esm

#UPDATE VERSION функция возвращает таблицу с аналогами по условиям

def slice\_tb(x, id\_):

sql\_query1 = 'WHERE diametrtruby = (%s) AND stenkatruby = (%s) AND kalibrelm = (%s) AND diametropravkimmmpm = (%s) ORDER BY mpmin DESC LIMIT 100'

new\_tbl = functions2.data\_query(sql\_query1, [float(i) for i in ff(x)], data = 'dash\_analogue')

new\_tbl['Источник данных'] = 'Данные со стана'

new\_tbl = new\_tbl[['mpmin', 'markastali', 'Источник данных', 'idpipesinuque']]

new\_tbl.rename(columns = {"mpmin": "Data", "idpipesinuque": "ID"},

inplace = True)

sql\_query2 = 'WHERE diametrtruby = (%s) AND stenkatruby = (%s) AND kalibrelm = (%s) AND diametropravkimmmpm = (%s) ORDER BY ctid DESC LIMIT 100'

new\_tbl2 = functions2.data\_query(sql\_query2, [float(i) for i in ff(x)], data = 'dash\_pipes')

new\_tbl2['Источник данных'] = 'Фактические настройки'

new\_tbl2['Data'] = None

new\_tbl2 = new\_tbl2[['Data', 'markastali', 'Источник данных', 'id']]

new\_tbl2.rename(columns = {"id": "ID"},

inplace = True)

sql\_query3 = 'WHERE vtz\_report2\_1 = (%s) AND vtz\_report2\_3 = (%s) AND vtz\_report1\_1 = (%s) AND vtz\_report1\_2 = (%s) ORDER BY datetime DESC LIMIT 100'

new\_tbl3 = functions2.data\_query(sql\_query3, ff(x), data = 'data\_report')

new\_tbl3 = new\_tbl3[(new\_tbl3['scenarioid'] != id\_)]

new\_tbl3['Источник данных'] = 'Предварительные расчеты'

new\_tbl3.rename(columns = {"datetime": "Data", "scenarioid": "ID", 'vtz\_report1\_3': 'markastali'},

inplace = True)

new\_tbl3 = new\_tbl3[['Data', 'markastali', 'Источник данных', 'ID']]

sql\_query4 = 'WHERE diametrtruby = (%s) AND stenkatruby = (%s) AND kalibrelm = (%s) AND diametropravkimmmpm = (%s) ORDER BY timestamp DESC LIMIT 100'

new\_tbl4 = functions2.data\_query(sql\_query4, [float(i) for i in ff(x)], data = 'integrazaya')

new\_tbl4['Источник данных'] = 'Данные со стана (интеграция)'

new\_tbl4 = new\_tbl4[['timestamp', 'markastali', 'Источник данных', 'id']]

new\_tbl4.rename(columns = {"timestamp": "Data", "id": "ID"}, inplace = True)

Table = pd.concat([new\_tbl, new\_tbl2, new\_tbl3, new\_tbl4])

Table['Data'] = Table['Data'].astype('datetime64[ns]').dt.round('S').dt.strftime('%Y-%m-%d %H:%M:%S')

return Table

#app = dash.Dash(\_\_name\_\_, external\_stylesheets=[dbc.themes.FLATLY], prevent\_initial\_callbacks=True)

app = dash.Dash(\_\_name\_\_, prevent\_initial\_callbacks=True)

server = app.server

#ТАБЛИЦЫ ДЛЯ ВВОДА

TABLE\_01 = dash\_table.DataTable(

id='table-01',

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": True},],

data = display\_tb(df, df2 = pd.DataFrame())[0].to\_dict('records'),

cell\_selectable = True,

#selected\_columns=[],

style\_as\_list\_view=True,

style\_cell={

'whiteSpace': 'normal',

#'height': 'auto',

#'height': '13px',

#'font-size': '12px',

#'minWidth': '150%', 'width': '150%', 'maxWidth': '150%',

# 'height': 5,

#'overflow': 'hidden',

#'textOverflow': 'ellipsis',

# 'maxWidth': 0,

# "backgroundColor": "#58606A",

# 'color': 'white',

},

style\_data\_conditional=[

{

'if': {'row\_index': 4},

'backgroundColor': "#F5F5F5",

'font-weight': 'bold',

}],

dropdown\_conditional=[{

'if': {

'column\_id': 'Значение',

#'row\_index': 2,

'filter\_query': '{Параметр} eq "Марка стали"'

},

'options': [

{'label': i, 'value': i}

for i in list\_marki\_1

]

},

{

'if': {

'column\_id': 'Значение',

#'row\_index': 2,

'filter\_query': '{Параметр} eq "Калибр, мм"'

},

'options': [

{'label': i, 'value': i}

for i in list\_kalibr\_1

]

}],

#editable=[{'Значение': True, 'Параметр': False}],

style\_cell\_conditional=[

# {

# 'if': {'column\_id': c},

# 'width': '150%',

# 'textAlign': 'left'

# } for c in ['Параметр', 'Значение']

{'if': {'column\_id': 'Параметр'},

'width': '70%', 'textAlign': 'left'},

{'if': {'column\_id': 'Значение'},

'width': '30%', 'textAlign': 'left'},

],

style\_header={

#"backgroundColor": "#353A40", #292D32

'textAlign': 'left',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",

},

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 20px'},

{"selector": ".Select-menu-outer", "rule": "display: block !important"},],

)

def table2(df):

TABLE\_02 = dash\_table.DataTable(

id='table-02',

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": True, 'type': 'numeric'},],

data = df.to\_dict('records'),

cell\_selectable = True,

#selected\_columns=[],

style\_as\_list\_view=True,

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

#'font-size': '12px'

},

style\_data\_conditional=[

{'if': {'row\_index': c},

'backgroundColor': "#c7e5f3",

#'font-weight': 'bold',

} for c in [5,6,7,8,9,10]],

#editable=[{'Значение': True, 'Параметр': False}],

style\_cell\_conditional=[

# {

# 'if': {'column\_id': c},

# #'width': '150%',

# 'textAlign': 'left'

# } for c in ['Параметр', 'Значение'],

{'if': {'column\_id': 'Параметр'},

'width': '70%', 'textAlign': 'left'},

{'if': {'column\_id': 'Значение'},

'width': '30%', 'textAlign': 'left'},

],

style\_header={

#"backgroundColor": "#353A40", #292D32

'textAlign': 'left',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",

},

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 20px'}],

)

return TABLE\_02

TABLE\_03 = dash\_table.DataTable(

id='table-03',

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": True, 'type': 'numeric'},],

data = display\_tb(df, df2 = pd.DataFrame())[2].iloc[[0,1, 4,5, 8,9,12,13, 16, 17, 20,21,24,25, 28]].to\_dict('records'),

cell\_selectable = True,

#selected\_columns=[],

style\_as\_list\_view=True,

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

# 'font-size': '12px'

},

style\_data\_conditional=(

[{'if': {'row\_index': d},

'backgroundColor': "#F5F5F5",

'font-weight': 'bold',

} for d in [0, 2, 4, 6, 8, 10, 12]

]),

style\_cell\_conditional=([

{

'if': {'column\_id': c},

'textAlign': 'left'

} for c in ['Параметр', 'Значение']]

),

style\_header={

#"backgroundColor": "#353A40", #292D32

'textAlign': 'left',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",

},

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 20px'}],

#virtualization=True,

)

TABLE\_04 = dash\_table.DataTable(

id='table-04',

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": True, 'type': 'numeric'},],

data = display\_tb(df, df2 = pd.DataFrame())[3].iloc[[0,1,4,5, 8,9,12,13, 16, 17, 20,21,24,25, 28, 29, 32, 33]].to\_dict('record'),

cell\_selectable = True,

#selected\_columns=[],

style\_as\_list\_view=True,

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

# 'font-size': '12px'

},

style\_data\_conditional=(

[{'if': {'row\_index': d},

'backgroundColor': "#F5F5F5",

'font-weight': 'bold',

} for d in [0, 2, 4, 6, 8, 10, 12, 14, 16]

]),

style\_cell\_conditional=[

{

'if': {'column\_id': c},

'textAlign': 'left'

} for c in ['Параметр', 'Значение']

],

style\_header={

#"backgroundColor": "#353A40", #292D32

'textAlign': 'left',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",

},

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 20px'}],

# fixed\_rows={'headers': True},

)

pg1 = choose\_table\_create(functions2.sql\_data\_LIMIT500(limit = 3))

col\_names = ['Сценарий', 'Дата создания', 'Калибр, мм', 'Код оправки',

'Марка стали', 'Диаметр трубы, мм', 'Толщина стенки, мм' ]

#функия для создания таблицы с выбором аналогов

def table\_modal(pg, col\_names):

chose\_table = dash\_table.DataTable(

id='select\_table',

columns= [{"name": j, "id": i}

for i,j in zip(pg.columns, col\_names)],

data = pg.to\_dict('record'),

row\_selectable = 'single',

filter\_action="native",

#page\_action='native',

page\_current= 0,

page\_size= 10,

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

'overflow': 'hidden',

'font-size': '12px'},

style\_cell\_conditional=[{'if': {'column\_id': 'scenarioid'},'display': 'none'}],

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 15px'}])

return chose\_table

#функия для создания таблицы с зазорами и оборотами

def zazor\_table(df):

zazor\_table = dash\_table.DataTable(

id='zazor\_table',

columns= [{"name": i, "id": i}

for i in df.columns],

data = df.to\_dict('record'),

style\_cell={

'whiteSpace': 'normal',

'height': 'auto',

'overflow': 'hidden',

'textAlign': 'center',

'font-size': '12px'},

style\_cell\_conditional=[{'if': {'row\_index': 'odd'},

'backgroundColor': "#F5F5F5",},

],

style\_header={

'textAlign': 'center',

'font-weight': 'bold',

'backgroundColor': "#F5F5F5",},

css=[{'selector': '.dash-spreadsheet tr', 'rule': 'height: 14px'}])

return zazor\_table

toast\_01 = dbc.Toast(

[TABLE\_01],

header="1. ВВОДНЫЕ ДАННЫЕ", id = 'toast01', style={'margin-bottom': '5px'}

)

toast\_02 = dbc.Toast(

#children = table2(tb2.iloc[0:5]),

children = table2(display\_tb(df, df2 = pd.DataFrame())[1].iloc[0:5]),

#[TABLE\_02],

#[table2(tb2)],

header="2. ПРОШИВНОЙ СТАН", id = 'toast02', #style={"maxWidth": "900px"}

)

toast\_03 = dbc.Toast(

[TABLE\_03],

header="3. НЕПРЕРЫВНЫЙ СТАН", id = 'toast03', #style={"maxWidth": "900px"}

)

toast\_04 = dbc.Toast(

[TABLE\_04],

header="4. ИЗВЛЕКАТЕЛЬНО-КАЛИБРОВОЧНЫЙ СТАН", id = 'toast04', #style={"maxWidth": "900px"}

)

style\_1={

'margin-right': '2px',

}

#### APP #####

app.layout = html.Div([

dcc.Location(id='url', refresh=False),

html.Div(id='page-content')

])

#### PAGE 1 #####

####PAGE 1 BUTTONS#####

load\_report = html.Div([dbc.DropdownMenu(

[dbc.DropdownMenuItem("Отчет МПМ", id="item-MPM"), dbc.DropdownMenuItem("Отчет ПС", id="item-PS")],

label="Технологическая карта", color="primary", bs\_size="sm",

group=True, id = 'dropdown1'),

Download(id="download"),])

report= html.Div([dbc.Button(

"Расчет", color="primary", size="sm", id = 'model'),

dbc.Modal(

[

dbc.ModalHeader("ПРЕДУПРЕЖДЕНИЕ!"),

dbc.ModalBody([html.Div("Для запуска расчета, пожалуйста, заполните все данные!"),

html.Br(),

html.Div("Если ВСЕ данные заполнены, то, пожалуйста, еще раз проверьте введенные параметры (толщину стенки, оправку МПМ, калибр и диаметр трубы, наличие резов)")]),

dbc.ModalFooter(

dbc.Button("Закрыть", id="close", className="ml-auto")

),

],

style={"margin-right": '0px !important', 'padding-right': '0px !important'},

id="modal",

),])

load= html.Div([dbc.Button(

"Выбрать расчет", color="primary", size="sm", id = 'load'),

dbc.Modal(

[

dbc.ModalHeader("Выберите расчет, который Вы хотите загрузить"),

dbc.ModalBody(table\_modal(pg1, col\_names)),

#dbc.ModalBody('oppp'),

dbc.ModalFooter([

dbc.Button("Загрузить", id="ok", className="ml-auto", color="primary"),

dbc.Button("Удалить", id="del", className="ml-auto", color="primary"),

dbc.Button("Отмена", id="return", className="ml-auto", color="primary"),]

),

],

style={"margin-right": '0px !important', 'padding-right': '0px !important'},

id="modal2", size="xl",)

])

garbage= dbc.Button(

"Очистить данные", color="primary", size="sm", id = 'clear')

choose\_analog\_btn= dbc.Button(

"Выбрать аналог", color="primary", size="sm", id = 'choose\_analog\_btn', style={'display': 'none'})

page= dbc.Button(

"Идеальные трубы", color="primary", size="sm", id = 'page', href='/table')

page\_3= dbc.Button(

"Реальные трубы", color="primary", size="sm", id = 'page\_3', href='/realpipes') #переход на страницу 3

group1 = dbc.ButtonGroup(

[load, report, garbage, load\_report, choose\_analog\_btn, page, page\_3

], id = 'buttongrp\_1')

# неизменные данные для графика (ocь X) - номера клети

x\_data = [1,2,3,4,5,6,7]

###SRYLE FOR UP-D TABLES

style\_table1\_2=[

{

'if': {'row\_index': 5},

'backgroundColor': "#F5F5F5",

'font-weight': 'bold',

}]

style\_table3=([

{'if': {'row\_index': c},

'backgroundColor': "#c7e5f3",

# 'display': 'none',

#'visibility': 'hidden'

#'font-weight': 'bold',

} for c in [2,3,6,7,10,11,14,15,18,19,22,23, 26, 27]] +

[{'if': {'row\_index': d},

'backgroundColor': "#F5F5F5",

'font-weight': 'bold',

} for d in [0, 4, 8, 12, 16, 20, 24]

])

style\_table4=([

{'if': {'row\_index': c},

'backgroundColor': "#c7e5f3",

#'font-weight': 'bold',

} for c in [2,3,6,7,10,11,14,15,18,19,22,23, 26, 27, 30, 31, 34, 35]] +

[{'if': {'row\_index': d},

'backgroundColor': "#F5F5F5",

'font-weight': 'bold',

} for d in [0, 4, 8, 12, 16, 20, 24, 28, 32]

])

style\_table1\_1=[

{

'if': {'row\_index': 4},

'backgroundColor': "#F5F5F5",

'font-weight': 'bold',

}]

#### page 1 load analog button ####

save\_ideal\_btn1 = html.Div([dbc.Button('Сохранить аналог в качестве идеальной трубы', id='save\_ideal\_btn\_pg1', n\_clicks=0, size="sm", color="primary",),

dbc.Modal(

[

dbc.ModalHeader(),

dbc.ModalBody([

html.P(id = 'load\_resultpg1\_1'),

]),

dbc.ModalFooter([

dbc.Button("Подтвердить сохранение", id="oksave\_page1", className="ml-auto"),

dbc.Button("Отмена", id="rejectsave\_page1", className="ml-auto"),]

),

],

style={"margin-right": '0px !important', 'padding-right': '0px !important'},

id="modal\_analog2\_page1", #size="md",

),

dbc.Modal(

[

dbc.ModalHeader(),

dbc.ModalBody([

html.P(id = 'load\_resultpg1\_2'),

]),

dbc.ModalFooter([

dbc.Button("Закрыть", id="closeanalogmodal\_page1", className="ml-auto"),]

),

],

style={"margin-right": '0px !important', 'padding-right': '0px !important'},

id="modal\_analog\_page1", #size="xl",

),

], style= {'display':'none'}, id = 'save\_analogpipes\_div')

#### PAGE 1 LAYOUT#####

page\_1\_layout = dbc.Container([

dbc.Row(html.P()),

group1,

dbc.Row(html.P()),

dbc.Row([

dbc.Col(html.Div([toast\_01, toast\_02]), style = style\_1,),

dbc.Col(toast\_03, style = style\_1,),

dbc.Col(toast\_04),

],

no\_gutters = True),

html.P(),

dbc.Row([dbc.Col([

html.Div(

dbc.Card(dcc.Graph(#figure = graph\_1(x\_data, y\_data),

style = {'height': "22vw"},

id = 'graph\_1'

), body = True, color="light"), id ='hide', style={'display': 'none'}),

html.P(),

html.Div(dash\_table.DataTable(id='table\_analog'), id = 'table\_from\_sql',),

html.P(),

save\_ideal\_btn1

], width = 6),

dbc.Col([html.Div(id = 'div\_analog\_table', style={'height': '100%'}),

], width = 6,)

]),

html.P(),

dcc.Store(id = 'store'),

dcc.Store(id = 'store2'),

html.Div(id = 'non\_existent', children =

dcc.Loading(type='graph', children=html.Div(

id='loading-hidden-div', children=None, style={'display': 'none'}),

id = 'loading\_state', fullscreen = False)

),

], fluid = True)

#### PAGE 2 #####

#### PAGE 2 BUTTONS #####

page\_2= dbc.Button("Рассчитать зазоры и скорости", color="primary", size="sm", id = 'page\_2', href='/') #переход на страницу

add\_2 = dbc.Button('Добавить', id='editing-rows-button', n\_clicks=0, size="sm", color="primary",)

# add\_2 = dcc.Loading(id="loading-2page\_1",

# children=[dbc.Button('Добавить', id='editing-rows-button', n\_clicks=0, size="sm", color="primary",)],

# type="circle",)

#save\_2 = dbc.Button('Сохранить', id='save\_to\_postgres', n\_clicks=0, size="sm", color="primary",)

save\_2 = dcc.Loading(id="loading-2page\_2",

children=[dbc.Button('Сохранить', id='save\_to\_postgres', n\_clicks=0, size="sm", color="primary",)],

type="circle",)

del\_2 = dbc.Button('Удалить', id='del\_to\_postgres', n\_clicks=0, size="sm", color="primary",)

# del\_2 = dcc.Loading(id="loading-2page\_3",

# children=[dbc.Button('Удалить', id='del\_to\_postgres', n\_clicks=0, size="sm", color="primary",)],

# type="circle",)

group2 = dbc.ButtonGroup([add\_2, save\_2, del\_2, page\_2, page\_3], id = 'buttongrp\_2')

#PAGE 2 LAYOUT#####

page\_2\_layout = dbc.Container([

dcc.Interval(id='interval\_pg', interval=86400000\*7,n\_intervals=0),#

html.P(),

group2,

html.P(),

dbc.Row(

[dbc.Col(html.Div(id='postgres\_datatable',

children=table\_1(data=pd.DataFrame(),

row=None,data\_name=data\_name)[0],style={'display': 'none'}),width = 12)]),

html.Button('index', id='index', n\_clicks=-1,style={'display': 'none'}),

html.Div(id='placeholder',children=[]),

dcc.Interval(id='interval', interval=1000000),

html.P(),

dbc.Row(

[

dbc.Col(html.Div(id='postgres\_datatable2',children=table\_2,style={'display': 'none'}), width = 5),

dbc.Col(html.Div(id='out-plot',children=plot\_1,style={'display': 'none'}), width = 7),

]

),],fluid = True)

#### PAGE 3 #####

#### PAGE 3 BUTTONS #####

kalibrelm = markastali = diametropravkimmmpm = diametrtruby = stenkatruby = []

renew = dbc.Button('Обновить данные', id='renew\_data', n\_clicks=0, size="sm", color="primary",)

# renew = dbc.Button('Обновить', id='renew\_data', n\_clicks=0, size="sm", color="primary",)

condition\_load = html.Div([dbc.Button('Выгрузить данные по условию', id='condition\_load\_btn', n\_clicks=0, size="sm", color="primary",),

dbc.Modal(

[

dbc.ModalHeader("Выберите данные, которые Вы хотите отобразить"),

dbc.ModalBody([

html.P(),

dbc.Row([

dbc.Col([

html.Div([html.Div("Калибр\_ELM"),

dcc.Dropdown(

options=[{'label': i, 'value': i} for i in kalibrelm],

value= None,

multi=True, id = 'selector\_kalibrelm')

]),

html.P(),

html.Div([html.Div("Марка стали"),

dcc.Dropdown(

options=[{'label': i, 'value': i} for i in markastali],

value= None,

multi=True, id = 'selector\_markastali')

]),

html.P(),

dbc.Row([html.Div('Период времени, за который будут выгружены данные'),

dbc.Col([ #html.Div('C периода'),

dcc.Input(type= 'date', id = 'date\_from')

]),

dbc.Col([ #html.Div('По период'),

dcc.Input(type= 'date', id = 'date\_to')]),

]),

]),

dbc.Col([

html.Div([html.Div("Стенка трубы"),

dcc.Dropdown(

options=[{'label': i, 'value': i} for i in stenkatruby],

value= None,

multi=True, id = 'selector\_stenkatruby')

]),

html.P(),

html.Div([html.Div("Диаметр трубы"),

dcc.Dropdown(

options=[{'label': i, 'value': i} for i in diametrtruby],

value= None,

multi=True, id = 'selector\_diametrtruby')

]),

html.P(),

html.Div([html.Div("Диаметр оправки(мм) MPM"),

dcc.Dropdown(

options=[{'label': i, 'value': i} for i in diametropravkimmmpm],

value= None,

multi=True, id = 'selector\_diametropravkimmmpm')

]),

]),

html.P(),

html.P("Одновременно загружается не более 500 последних по дате труб, подходящих под выбранные условия!",

style={'color': 'red', 'font-weight': 'bold'})

])

]),

dbc.ModalFooter([

dbc.Button("Загрузить", id="ok\_page3", className="ml-auto"),

dbc.Button("Отмена", id="return\_page3", className="ml-auto"),]

),

],

style={"margin-right": '0px !important', 'padding-right': '0px !important'},

id="modal\_page3", size="xl",)])

save\_ideal\_btn = html.Div([dbc.Button('Сохранить как идеальную трубу', id='save\_ideal\_btn', n\_clicks=0, size="sm", color="primary",),

dbc.Modal(

[

dbc.ModalHeader(),

dbc.ModalBody([

html.P(id = 'load\_result'),

]),

dbc.ModalFooter([

dbc.Button("Подтвердить сохранение", id="oksave\_page3", className="ml-auto"),

dbc.Button("Отмена", id="rejectsave\_page3", className="ml-auto"),]

),

],

style={"margin-right": '0px !important', 'padding-right': '0px !important'},

id="modal\_analog2\_page3", #size="md",

),

dbc.Modal(

[

dbc.ModalHeader(),

dbc.ModalBody([

html.P(id = 'load\_result2'),

]),

dbc.ModalFooter([

dbc.Button("Закрыть", id="closeanalogmodal\_page3", className="ml-auto"),]

),

],

style={"margin-right": '0px !important', 'padding-right': '0px !important'},

id="modal\_analog\_page3", #size="xl",

),

])

group3 = dbc.ButtonGroup([condition\_load, renew, save\_ideal\_btn, page, page\_2], id = 'buttongrp\_3')

#CALLBACKS

###CONTROL CELLS FOR UNPUT##### запрещают ввод в определенные ячейки

@app.callback(

Output("table-01", "columns"),

Output("table-01", "dropdown\_conditional"),

[Input("table-01", "active\_cell"),

Input("table-01", "data"),

],

)

def tb01\_status(cell, data):

###new dropdowns lists###

new\_tbl = functions2.sql\_data\_LIMIT500(data = 'dash\_analogue', order\_id = 'mpmin', limit = 1500) #sql\_data(data = 'dash\_analogue')

new\_tbl = new\_tbl[['kalibrelm', 'markastali']]

#print(new\_tbl)

new\_tbl2 = functions2.sql\_data\_LIMIT500(data = 'dash\_pipes', limit = 1500) #sql\_data(data = 'dash\_pipes')

new\_tbl2 = new\_tbl2[['kalibrelm', 'markastali']]

#print(new\_tbl2)

new\_tbl3 = functions2.sql\_data\_LIMIT500(data = 'data\_report', order\_id = 'datetime', limit = 1500) #sql\_data()

#print(new\_tbl3)

new\_tbl3.rename(columns = {"vtz\_report1\_1": "kalibrelm", 'vtz\_report1\_3': 'markastali'},

inplace = True)

new\_tbl3 = new\_tbl3[['kalibrelm', 'markastali']]

#print(new\_tbl3)

new\_tbl4 = functions2.sql\_data\_LIMIT500(data = 'integrazaya', order\_id = 'timestamp', limit = 1500) #sql\_data(data = 'integrazaya')

new\_tbl4 = new\_tbl2[['kalibrelm', 'markastali']]

#print(new\_tbl4)

new\_tbl5 = pd.DataFrame()

new\_tbl5['kalibrelm'] = list\_kalibr['kalibr']

new\_tbl5['markastali'] = list\_marki['marki']

#print(new\_tbl5)

Table\_list = pd.concat([new\_tbl, new\_tbl2, new\_tbl3, new\_tbl4, new\_tbl5])

Table\_list['markastali'] = Table\_list['markastali'].astype('string')

Table\_list['kalibrelm'] = Table\_list['kalibrelm'].astype('float')

#Table\_list = Table\_list.dropna()

list\_marki\_2 = Table\_list['markastali'].unique()

list\_kalibr\_2 = Table\_list['kalibrelm'].unique()

dropdown\_conditional=[{

'if': {

'column\_id': 'Значение',

#'row\_index': 2,

'filter\_query': '{Параметр} eq "Марка стали"'

},

'options': [

{'label': i, 'value': i}

for i in list\_marki\_2

]

},

{

'if': {

'column\_id': 'Значение',

'filter\_query': '{Параметр} eq "Калибр, мм"'

},

'options': [

{'label': i, 'value': i}

for i in list\_kalibr\_2

]

}]

if len(data)>8:

list\_a = [4, 5]

else:

list\_a = [4]

#changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

if cell and cell['row'] !=None:

if cell['row'] in list\_a: #код оправки - не рададктируется

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": False},]

elif cell['row']==2 or cell['row']==3: #списки - марки стали и диаметр

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": True, 'presentation': 'dropdown'},]

else:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": True, 'type': 'numeric'},]

else:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": False},]

return columns, dropdown\_conditional

@app.callback(

Output("table-02", "columns"),

[Input("table-02", "active\_cell"),],

)

def tb02\_status(cell):

#changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

if cell and cell['row'] != None:

if cell['row'] in [5,6,7,8,9,10]:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": False},]

else:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": True, 'type': 'numeric'},]

else:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": False},]

return columns

@app.callback(

Output("table-03", "columns"),

[Input("table-03", "active\_cell"),

Input("table-03", "data"),

],

)

def tb03\_status(cell, data):

if len(data)>16:

list\_a = [0, 2,3,4, 6,7, 8, 10, 11, 12, 14, 15, 16, 18, 19, 20, 22, 23, 24, 26, 27]

else:

list\_a = [0, 2, 4, 6, 8, 10, 12]

#changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

if cell and cell['row'] !=None:

if cell['row'] in list\_a:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": False},]

else:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": True, 'type': 'numeric'},]

return columns

else:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": False},]

return columns

@app.callback(

Output("table-04", "columns"),

[Input("table-04", "active\_cell"),

Input("table-04", "data")

],

)

def tb04\_status(cell, data):

if len(data)>20:

list\_a = [0, 2,3,4, 6,7, 8, 10, 11, 12, 14, 15, 16, 18, 19, 20, 22, 23, 24,

26, 27, 28, 30, 31, 32, 34, 35]

else:

list\_a = [0, 2, 4, 6, 8, 10, 12, 14, 16]

changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

if 'table-04.active\_cell' in changed\_id:

if cell['row'] in list\_a:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": False},]

else:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": True, 'type': 'numeric'},]

return columns

else:

columns= [

{'name': 'Параметр', 'id': 'Параметр', "editable": False},

{'name': 'Значение', 'id': 'Значение', "editable": False},]

return columns

@app.callback(

Output("loading\_state", "fullscreen"),

Output("loading\_state", "style"),

Input("model", "n\_clicks"),

Input("ok", "n\_clicks"),

Input("choose\_analog\_btn", "n\_clicks"),

)

def loading\_state(model, ok, analog):

changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

if 'model' or "ok" or "analog" in changed\_id:

return True, {'display': 'block'}

else:

return False, {'display': 'none'}

@app.callback(

[Output("modal", "is\_open"), #всплывающее окно 1, появляется, если не хватает данных для модели

Output("table-02", "data"), #2 - данные для таблицы 2 - отображается на экране

Output("table-03", "data"), #3 - данные для таблицы 3 - отображается на экране

Output("table-03", "style\_data\_conditional"), #4 - цвет строк в таблице по условию

Output('hide', 'style'), #видимость графика со скоростями

Output("modal2", "is\_open"), #6 всплывающее окно 2, работа с Postgress - загрузка и удаление строк

Output("table-04", "data"), #7 - данные для таблицы 4 - отображается на экране

Output("table-04", "style\_data\_conditional"), #8 - цвет строк в таблице по условию

Output("graph\_1", "figure"), #9

Output("table\_from\_sql", "children"),#10

Output("select\_table", "data"), #таблица с аналогами (появляется по условию)

Output("table-01", "data"),#12

Output("table-01", "style\_data\_conditional"), #13 - цвет строк в таблице по условию

Output('store', 'data'), #текущий айди базы - (загруженный или посчитанный). В других случаях передается 0

Output("loading-hidden-div", "children"),

],

Input("table-01", "data"), #данные из вводных таблиц

Input("table-02", "data"), #данные из вводных таблиц

Input("table-03", "data"), #данные из вводных таблиц

Input("model", "n\_clicks"), #всплывающее окно 1 с предупреждением, если расчетн данные введены не полностью

Input("close", "n\_clicks"), #закрыть всплывающее окно 1

Input("load", "n\_clicks"), #всплывающее окно, позволяющее работа с БД

Input("ok", "n\_clicks"), #подтвержает выбор строки из базы и перенос данных на экран

Input("return", "n\_clicks"), #закрывает всплывающее окно без выбора или удаления

Input("del", "n\_clicks"), #удаление строки

Input("select\_table", "selected\_rows"), #выбор строки в таблице с аналогами. Отображает новый график, таблицу с зазорами и оборотами

Input("clear", "n\_clicks"),

Input("choose\_analog\_btn", "n\_clicks"), #кнопка для переноса анлога в расчет. Появляется при выборе строки в таблице аналога

Input('store2', 'data'), #данные из другой функции. Нужны для передачи новых зазоров,

#оборотов и т.д. при выборе и сохранении аналога как расчета

Input("table-04", "data"), #данные из вводных таблиц

Input('store', 'data'),

[State("modal", "is\_open"),

State("modal2", "is\_open"),

State("select\_table", "data")]

)

def calc(data1, data2, data3, n\_clicks, close\_clicks, load\_clicks, ok\_clicks,

close2\_clicks, del\_clicks, row\_sel\_tb, clear\_clicks, choose\_analog\_click, dt\_analog, data4, previous\_id,

is\_open, is\_open2, select\_table\_data):

changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

if 'model' in changed\_id:

check\_val\_tb1 = sum(x['Значение'] is not None for x in data1)

check\_val\_tb2 = sum(x['Значение'] is not None for x in data2)

check\_val\_tb3 = sum(x['Значение'] is not None for x in data3)

check\_val\_tb4 = sum(x['Значение'] is not None for x in data4)

##первоначальная -

if len(data2) == 11 and check\_val\_tb1 == 8 and check\_val\_tb2 == 11 and check\_val\_tb3 == 22 and check\_val\_tb4 == 27:

model\_df = pd.concat([pd.DataFrame(data1), pd.DataFrame(data2), pd.DataFrame(data3), pd.DataFrame(data4)])

model\_df = model\_df.reset\_index(drop = True)

model\_df = model\_df.reset\_index()

model\_data = df\_for\_model.merge(model\_df, left\_on='index', right\_on='index', how = 'left')

model\_data = model\_data[['name\_properties', 'Значение']]

model\_data = model\_data.rename(columns = {'Значение': 'new'}, inplace = False)

model\_data = model\_data.fillna(0)

#print(previous\_id)

#check data by condition

condition\_1 = "float(ff(model\_data[model\_data['name\_properties'] =='Калибр']['new'])) <= float(ff(model\_data[model\_data['name\_properties'] =='Оправка МПМ']['new']))"

condition\_2 = "float(ff(model\_data[model\_data['name\_properties'] =='Толщина трубы']['new'])) > 30"

condition\_3 = "float(ff(model\_data[model\_data['name\_properties'] =='Диаметр Трубы']['new'])) <100"

condition\_4 = "float(ff(model\_data[model\_data['name\_properties'] =='Диаметр Трубы']['new'])) >1000"

#условия, согласно которым введенные данные считаются неправильными и нуждаются в корректировке

if eval(condition\_1) == True or eval(condition\_2) == True or eval(condition\_3) == True or eval(condition\_4) == True:

#print('except by data condition')

list\_vals= [not is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update,

no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update]

return list\_vals

model\_data['new'] = model\_data['new'].astype('object')

try:

model\_ = model\_vtz.model\_v0(data\_properties\_VTZ=model\_data)

model\_ = model\_[['pred', 'name']]

model\_sql = head.merge(model\_, left\_on='name', right\_on='name', how = 'left')

model\_sql = model\_sql[['name', 'pred']].T

model\_sql.columns = model\_sql.iloc[0]

model\_sql = model\_sql.reset\_index(drop = True)

model\_sql = model\_sql.drop([0])

sss\_heattreatment=''

for i in range(0, model\_sql.shape[1]):

sss\_heattreatment=sss\_heattreatment + "%s,"

sss\_heattreatment=sss\_heattreatment[:-1]

for i in model\_sql.columns:

if i != 'datetime':

try:

model\_sql[i]=model\_sql[i].astype(str)

except:

model\_sql[i]=model\_sql[i].astype('object')

#по согласованию, всегда перезаписываем карту, если какие-либо расчеты были сделаны и расчет запустился заново

functions2.delete\_row\_by\_ID(del\_id=previous\_id)##удаляем предыдущую версию, оставляем только перезаписанную

#пересохраняем со старым ID

model\_sql['datetime']=model\_sql['datetime'].astype('datetime64[ns]') + timedelta(hours=3)

model\_sql['scenarioid'] = previous\_id

down\_status = functions2.save\_to\_sql(sql\_base = model\_sql, sss\_heattreatment = sss\_heattreatment)

#print(down\_status)

current\_id = ff(model\_sql['scenarioid'])

#print('resave', current\_id)

df\_select = functions2.data\_by\_ID(track\_id = current\_id)

df\_select = df\_select.round(1)

try:

df\_select['vtz\_report1\_6'] = round((df\_select['vtz\_report1\_6'].astype('float') \* df\_select['cuts'].astype('float')), 0)

#print('ok')

except:

df\_select['vtz\_report1\_6'] = df\_select['vtz\_report1\_6']

y\_data\_new = df\_select[['vtz\_report2\_155', 'vtz\_report2\_156', 'vtz\_report2\_157', 'vtz\_report2\_158',

'vtz\_report2\_159', 'vtz\_report2\_160', 'vtz\_report2\_161']].values.tolist()

y\_data\_float = list(np.float\_(y\_data\_new))

df\_select2 = df\_select[['vtz\_report2\_1', 'vtz\_report2\_3', 'vtz\_report1\_1', 'vtz\_report1\_2']]

check\_list = df\_select2.values.tolist()

list\_vals= [is\_open, display\_tb(df, df2 = df\_select)[1].to\_dict('records'),

display\_tb(df, df\_select)[2].to\_dict('records'), style\_table3,

{'display': 'block'}, no\_update,

display\_tb(df, df2 = df\_select)[3].to\_dict('records'), style\_table4, graph\_1(x\_data, y\_data\_float[0]), #no\_update,

create\_analog\_tabl(slice\_tb(check\_list, current\_id)),

no\_update, display\_tb(df, df2 = df\_select)[0].to\_dict('records'),

style\_table1\_2, current\_id, []]

except:

#print('except')

list\_vals= [not is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update,

no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update]

return list\_vals

elif len(data2) == 5 and check\_val\_tb1 == 7 and check\_val\_tb2 == 5 and check\_val\_tb3 == 8 and check\_val\_tb4 == 9:

model\_df = pd.concat([pd.DataFrame(data1), pd.DataFrame(data2), pd.DataFrame(data3), pd.DataFrame(data4)])

model\_df = model\_df.reset\_index(drop = True)

model\_df = model\_df.reset\_index()

model\_data = df\_for\_model.merge(model\_df, left\_on='index\_new', right\_on='index', how = 'left')

model\_data = model\_data[['name\_properties', 'Значение']]

model\_data = model\_data.rename(columns = {'Значение': 'new'}, inplace = False)

model\_data = model\_data.fillna(0)

model\_data.to\_excel("data\_input\_model.xlsx")

model\_data['new'] = model\_data['new'].astype('object')

try:

model\_ = model\_vtz.model\_v0(data\_properties\_VTZ=model\_data)

model\_ = model\_[['pred', 'name']]

model\_sql = head.merge(model\_, left\_on='name', right\_on='name', how = 'left')

model\_sql = model\_sql[['name', 'pred']].T

model\_sql.columns = model\_sql.iloc[0]

model\_sql = model\_sql.reset\_index(drop = True)

model\_sql = model\_sql.drop([0])

#print(model\_sql)

sss\_heattreatment=''

for i in range(0, model\_sql.shape[1]):

sss\_heattreatment=sss\_heattreatment + "%s,"

sss\_heattreatment=sss\_heattreatment[:-1]

for i in model\_sql.columns:

if i != 'datetime':

try:

model\_sql[i]=model\_sql[i].astype(str)

except:

model\_sql[i]=model\_sql[i].astype('object')

model\_sql['datetime']=model\_sql['datetime'].astype('datetime64') + timedelta(hours=3)

down\_status = functions2.save\_to\_sql(sql\_base = model\_sql, sss\_heattreatment = sss\_heattreatment)

#print(down\_status)

current\_id = ff(model\_sql['scenarioid'])

#print(current\_id)

df\_select = functions2.data\_by\_ID(track\_id = current\_id)

df\_select = df\_select.round(1)

try:

df\_select['vtz\_report1\_6'] = round((df\_select['vtz\_report1\_6'].astype('float') \* df\_select['cuts'].astype('float')), 0)

except:

df\_select['vtz\_report1\_6'] = df\_select['vtz\_report1\_6']

y\_data\_new = df\_select[['vtz\_report2\_155', 'vtz\_report2\_156', 'vtz\_report2\_157', 'vtz\_report2\_158',

'vtz\_report2\_159', 'vtz\_report2\_160', 'vtz\_report2\_161']].values.tolist()

y\_data\_float = list(np.float\_(y\_data\_new))

df\_select2 = df\_select[['vtz\_report2\_1', 'vtz\_report2\_3', 'vtz\_report1\_1', 'vtz\_report1\_2']]

check\_list = df\_select2.values.tolist()

list\_vals= [is\_open, display\_tb(df, df2 = df\_select)[1].to\_dict('records'),

display\_tb(df, df\_select)[2].to\_dict('records'), style\_table3,

{'display': 'block'}, no\_update,

display\_tb(df, df2 = df\_select)[3].to\_dict('records'), style\_table4, graph\_1(x\_data, y\_data\_float[0]), #no\_update,

create\_analog\_tabl(slice\_tb(check\_list, current\_id)),

no\_update, display\_tb(df, df2 = df\_select)[0].to\_dict('records'),

style\_table1\_2, current\_id, []]

except:

#print('except')

list\_vals= [not is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update,

no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update]

return list\_vals

else:

return not is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update,\

no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, []

elif 'close' in changed\_id:

return not is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update,\

no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, []

elif 'load' in changed\_id:

#при открытии вплывающего окна заново подгружаем данные из БД

pg1 = choose\_table\_create(functions2.sql\_data\_LIMIT500(data = 'data\_report', order\_id = 'datetime', limit = 300))

return is\_open, no\_update, no\_update, no\_update, no\_update, not is\_open2,\

no\_update, no\_update, no\_update, no\_update, pg1.to\_dict('record'), no\_update, no\_update, no\_update,\

[]

elif 'return' in changed\_id:

return no\_update, no\_update, no\_update, no\_update, no\_update, not is\_open2,\

no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, []

elif 'ok' in changed\_id:

pg = choose\_table\_create(functions2.sql\_data\_LIMIT500(data = 'data\_report', order\_id = 'datetime', limit = 300))

current\_id = ff(pd.DataFrame(select\_table\_data)['scenarioid'][row\_sel\_tb]) #текущий айди, который будет передаваться между функциями

#print(row\_sel\_tb)

#print(current\_id)

df\_select = functions2.data\_by\_ID(track\_id = ff(pg['scenarioid'][row\_sel\_tb]))

y\_data\_new = df\_select[['vtz\_report2\_155', 'vtz\_report2\_156', 'vtz\_report2\_157', 'vtz\_report2\_158',

'vtz\_report2\_159', 'vtz\_report2\_160', 'vtz\_report2\_161']].values.tolist()

y\_data\_float = list(np.float\_(y\_data\_new))

df\_select2 = df\_select[['vtz\_report2\_1', 'vtz\_report2\_3', 'vtz\_report1\_1', 'vtz\_report1\_2']]

check\_list = df\_select2.values.tolist()

#print(check\_list)

try:

df\_select['vtz\_report1\_6'] = round((df\_select['vtz\_report1\_6'].astype('float') \* df\_select['cuts'].astype('float')), 0)

except:

df\_select['vtz\_report1\_6'] = df\_select['vtz\_report1\_6']

list\_vals= [is\_open, display\_tb(df, df2 = df\_select)[1].to\_dict('records'),

display\_tb(df, df2 = df\_select)[2].to\_dict('records'), style\_table3,

{'display': 'block'}, not is\_open2,

display\_tb(df, df2 = df\_select)[3].to\_dict('records'), style\_table4,

graph\_1(x\_data, y\_data\_float[0]),

create\_analog\_tabl(slice\_tb(check\_list, current\_id)),

no\_update, display\_tb(df, df2 = df\_select)[0].to\_dict('records'),

style\_table1\_2,

current\_id, []]

return list\_vals

elif 'del' in changed\_id:

#удаляем строку по айди

if row\_sel\_tb != None:

pg = choose\_table\_create(functions2.sql\_data\_LIMIT500(data = 'data\_report', order\_id = 'datetime', limit = 600))

#print(pg['scenarioid'][row\_sel\_tb])

#print(ff(pg['scenarioid'][row\_sel\_tb]))

functions2.delete\_row\_by\_ID(del\_id=ff(pg['scenarioid'][row\_sel\_tb]))

return is\_open, no\_update, no\_update, no\_update, no\_update, not is\_open2,\

no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, []

elif 'clear' in changed\_id:

current\_id = '0'

style3 = (

[{'if': {'row\_index': d},

'backgroundColor': "#F5F5F5",

'font-weight': 'bold',

} for d in [0, 2, 4, 6, 8, 10, 12]])

style4 =(

[{'if': {'row\_index': d},

'backgroundColor': "#F5F5F5",

'font-weight': 'bold',

} for d in [0, 2, 4, 6, 8, 10, 12, 14, 16]

])

return is\_open, display\_tb(df, df2 = pd.DataFrame())[1].iloc[0:5].to\_dict('records'),\

display\_tb(df, df2 = pd.DataFrame())[2].iloc[[0,1,4,5,8,9,12,13,16,17,20,21,24,25,28]].to\_dict('records'),\

style3, {'display': 'none'}, no\_update,\

display\_tb(df, df2 = pd.DataFrame())[3].iloc[[0,1,4,5,8,9,12,13,16,17,20,21,24,25,28,29,32,33]].to\_dict('record'),\

style4, no\_update, dash\_table.DataTable(id='table\_analog'), no\_update,\

display\_tb(df, df2 = pd.DataFrame())[0].to\_dict('records'), style\_table1\_1, current\_id, []

elif 'choose\_analog\_btn' in changed\_id:

#пересохраняем аналог, как новые данные (меняем в старых данных зазоры, обороты, скорости)

df\_select\_analog = pd.DataFrame.from\_dict(dt\_analog[0])

#print(df\_select\_analog['scenarioid'])

df\_select\_analog.loc[0, ['vtz\_report2\_43', 'vtz\_report2\_44', 'vtz\_report2\_45', 'vtz\_report2\_46',

'vtz\_report2\_47', 'vtz\_report2\_48', 'vtz\_report2\_49',

'vtz\_report2\_106', 'vtz\_report2\_107', 'vtz\_report2\_108', 'vtz\_report2\_109',

'vtz\_report2\_110', 'vtz\_report2\_111', 'vtz\_report2\_112', 'vtz\_report2\_113',

'vtz\_report2\_114']] = dt\_analog[1]

df\_select\_analog.loc[0, ['vtz\_report2\_36', 'vtz\_report2\_37', 'vtz\_report2\_38', 'vtz\_report2\_39',

'vtz\_report2\_40', 'vtz\_report2\_41', 'vtz\_report2\_42',

'vtz\_report2\_95', 'vtz\_report2\_96', 'vtz\_report2\_97', 'vtz\_report2\_98',

'vtz\_report2\_99', 'vtz\_report2\_100', 'vtz\_report2\_101', 'vtz\_report2\_102',

'vtz\_report2\_103']] = dt\_analog[2]

df\_select\_analog.loc[0, ['vtz\_report2\_155', 'vtz\_report2\_156', 'vtz\_report2\_157', 'vtz\_report2\_158',

'vtz\_report2\_159', 'vtz\_report2\_160', 'vtz\_report2\_161']] = dt\_analog[4]

df\_select\_analog.loc[0, ['vtz\_report2\_164', 'vtz\_report2\_165', 'vtz\_report2\_166', 'vtz\_report2\_167',

'vtz\_report2\_168', 'vtz\_report2\_169', 'vtz\_report2\_170']] = dt\_analog[5]

#пересчитываем новое время и id

now = datetime.now() + timedelta(hours=3)

df\_select\_analog["datetime"] = now.strftime("%Y-%m-%d %H:%M:%S")

# df\_select\_analog['scenarioid'] = str(ff(df\_select\_analog["vtz\_report2\_1"])) + "\_" + \

# str(ff(df\_select\_analog["vtz\_report2\_3"])) + "\_" + \

# str(ff(df\_select\_analog["vtz\_report1\_12"])) + "\_" + \

# str(ff(df\_select\_analog["vtz\_report1\_1"])) + "\_" +\

# str(ff(df\_select\_analog["vtz\_report1\_3"])) + "\_" +\

# str(ff(df\_select\_analog["datetime"]))

#диаметр, стенка, оправка, калибр, марка, дата

df\_select\_analog['source'][0] = str(dt\_analog[3]) #добавляем источник

df\_select\_analog['vtz\_report2\_130']=round(701.6601671630542-0.4069175131244827\*\

float(df\_select\_analog['vtz\_report2\_129'])+\

0.0441577157201906\*float(df\_select\_analog['vtz\_report2\_155'])+\

0.04415771572019076\*float(df\_select\_analog['vtz\_report2\_156'])+\

0.008953055135952077\*float(df\_select\_analog['vtz\_report2\_157'])+\

0.0015132335923520382\*float(df\_select\_analog['vtz\_report2\_158'])+\

0.004739170513118333\*float(df\_select\_analog['vtz\_report2\_159'])+\

0.0672449789942485\*float(df\_select\_analog['vtz\_report2\_160'])+\

0.013632662532926467\*float(df\_select\_analog['vtz\_report2\_161']),0)

sss\_heattreatment=''

for i in range(0, df\_select\_analog.shape[1]):

sss\_heattreatment=sss\_heattreatment + "%s,"

sss\_heattreatment=sss\_heattreatment[:-1]

for i in df\_select\_analog.columns:

if i != 'datetime':

try:

df\_select\_analog[i]=df\_select\_analog[i].astype(str)

except:

df\_select\_analog[i]=df\_select\_analog[i].astype('object')

df\_select\_analog['datetime']=df\_select\_analog['datetime'].astype('datetime64')

previous\_id = previous\_id ##предыдущий айдишник - рассчитанный или загруженный

#print(previous\_id)

df\_select\_analog2 = df\_select\_analog[['vtz\_report2\_1', 'vtz\_report2\_3', 'vtz\_report1\_1', 'vtz\_report1\_2']]

check\_list = df\_select\_analog2.values.tolist()

current\_id = ff(df\_select\_analog['scenarioid'])

#print(previous\_id, current\_id)

y\_data\_new = df\_select\_analog[['vtz\_report2\_155', 'vtz\_report2\_156', 'vtz\_report2\_157', 'vtz\_report2\_158',

'vtz\_report2\_159', 'vtz\_report2\_160', 'vtz\_report2\_161']].values.tolist()

y\_data\_float = list(np.float\_(y\_data\_new))

functions2.delete\_row\_by\_ID(del\_id=previous\_id)##удаляем предыдущую версию, оставляем только перезаписанную

down\_status = functions2.save\_to\_sql(sql\_base = df\_select\_analog, sss\_heattreatment = sss\_heattreatment)

#print(down\_status)

list\_vals= [is\_open, display\_tb(df, df2 = df\_select\_analog)[1].to\_dict('records'),

display\_tb(df, df2 = df\_select\_analog)[2].to\_dict('records'), style\_table3,

{'display': 'block'}, is\_open2,

display\_tb(df, df2 = df\_select\_analog)[3].to\_dict('records'), style\_table4,

graph\_1(x\_data, y\_data\_float[0]),

create\_analog\_tabl(slice\_tb(check\_list, current\_id)),

no\_update, display\_tb(df, df2 = df\_select\_analog)[0].to\_dict('records'),

style\_table1\_2,

current\_id, []]

return list\_vals

else:

return is\_open, no\_update, no\_update, no\_update, no\_update, no\_update,\

no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, []

@app.callback(

Output("div\_analog\_table", "children"), #таблица с оборотами и зазорами

#Output("clear", "n\_clicks"),

Output('hide', 'children'), #обновление графика (добавляем аналоги)

Output('choose\_analog\_btn', 'style'), #видимость кнопки с возможностью переноса анлога

Output('store2', 'data'),#данные для передачи в другую функцию (выбранныя строка, обороты, зазаоры и т.д.)

Output('save\_analogpipes\_div', 'style'),

[

#Input("table\_analog", "derived\_virtual\_selected\_rows"), #выбор строки

Input("table\_analog", "selected\_rows"), #выбор строки

Input("table\_analog", "data"),

Input('store', 'data')], #айди ткущей строки из БД

#Input("clear", "n\_clicks"),

)

def update\_new\_tb(row\_out, table\_data, data):

#функция, возвращающая таблицу с оборотами и зазорами аналога + обновляющая график (красная линия - аналог)

if row\_out:

#print(row\_out)

if row\_out==[]:

return dash\_table.DataTable(id='zazor\_table'), no\_update, {'display': 'none'}, None, {'display': 'none'}

elif row\_out!=[]:

df\_select = functions2.data\_by\_ID(track\_id = data)

y\_data\_new = df\_select[['vtz\_report2\_155', 'vtz\_report2\_156', 'vtz\_report2\_157', 'vtz\_report2\_158',

'vtz\_report2\_159', 'vtz\_report2\_160', 'vtz\_report2\_161']].values.tolist()

y\_data\_float = list(np.float\_(y\_data\_new))

Kalibr = ff(functions2.data\_by\_ID(data)['vtz\_report2\_6'])

diam\_fact\_mpm\_new = functions2.data\_by\_ID(data)[['vtz\_report2\_29', 'vtz\_report2\_30', 'vtz\_report2\_31',

'vtz\_report2\_32', 'vtz\_report2\_33',

'vtz\_report2\_34', 'vtz\_report2\_35', 'vtz\_report2\_85',

'vtz\_report2\_86', 'vtz\_report2\_87', 'vtz\_report2\_88',

'vtz\_report2\_89', 'vtz\_report2\_90', 'vtz\_report2\_91',

'vtz\_report2\_92', 'vtz\_report2\_93']]

#по нужной строке из созданной таблице берем айди (не отображ на экране)

if table\_data[int(row\_out[0])].get('Источник данных') == 'Данные со стана' or table\_data[int(row\_out[0])].get('Источник данных') == 'Данные со стана (интеграция)':

#берем уникальную строку по дате + айди

if table\_data[int(row\_out[0])].get('Источник данных') == 'Данные со стана':

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('Data'),

data = 'dash\_analogue', id\_ = "mpmin")

dt = dt[dt['idpipesinuque'] == table\_data[int(row\_out[0])].get('ID')]

else:

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('ID'), data = 'integrazaya', id\_ = "id")

#диаметры, зазоры и обороты аналога для отображения и пересчета

diam\_fact\_mpm = dt[['diametrvalkovmmklet1mpm',

'diametrvalkovmmklet2mpm', 'diametrvalkovmmklet3mpm',

'diametrvalkovmmklet4mpm', 'diametrvalkovmmklet5mpm',

'diametrvalkovmmklet6mpm', 'diametrvalkovmmklet7mpm',

'diametrvalkovmmklet1esm', 'diametrvalkovmmklet2esm',

'diametrvalkovmmklet3esm', 'diametrvalkovmmklet4esm',

'diametrvalkovmmklet5esm', 'diametrvalkovmmklet6esm',

'diametrvalkovmmklet7esm', 'diametrvalkovmmklet8esm',

'diametrvalkovmmklet9esm']]

zazor = dt[['zazorvalkovmmklet1mpm', 'zazorvalkovmmklet2mpm',

'zazorvalkovmmklet3mpm', 'zazorvalkovmmklet4mpm',

'zazorvalkovmmklet5mpm', 'zazorvalkovmmklet6mpm',

'zazorvalkovmmklet7mpm', 'zazorvalkovmmklet1esm',

'zazorvalkovmmklet2esm', 'zazorvalkovmmklet3esm',

'zazorvalkovmmklet4esm', 'zazorvalkovmmklet5esm',

'zazorvalkovmmklet6esm', 'zazorvalkovmmklet7esm',

'zazorvalkovmmklet8esm', 'zazorvalkovmmklet9esm']]

oborot = dt[['skorostvalkovklet1obminmpm',

'skorostvalkovklet2obminmpm', 'skorostvalkovklet3obminmpm',

'skorostvalkovklet4obminmpm', 'skorostvalkovklet5obminmpm',

'skorostvalkovklet6obminmpm', 'skorostvalkovklet7obminmpm',

'skorostkletiobminklet1esm', 'skorostkletiobminklet2esm',

'skorostkletiobminklet3esm', 'skorostkletiobminklet4esm',

'skorostkletiobminklet5esm', 'skorostkletiobminklet6esm',

'skorostkletiobminklet7esm', 'skorostkletiobminklet8esm',

'skorostkletiobminklet9esm']]

elif table\_data[int(row\_out[0])].get('Источник данных') == 'Фактические настройки': #по источнику данных определяем нужну таблицу в Postgress

#берем уникальную строку по дате + айди

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('ID'),

data = 'dash\_pipes', id\_ = "id")

diam\_fact\_mpm = dt[['diametrvalkovmmklet1mpm',

'diametrvalkovmmklet2mpm', 'diametrvalkovmmklet3mpm',

'diametrvalkovmmklet4mpm', 'diametrvalkovmmklet5mpm',

'diametrvalkovmmklet6mpm', 'diametrvalkovmmklet7mpm',

'diametrvalkovmmklet1esm', 'diametrvalkovmmklet2esm',

'diametrvalkovmmklet3esm', 'diametrvalkovmmklet4esm',

'diametrvalkovmmklet5esm', 'diametrvalkovmmklet6esm',

'diametrvalkovmmklet7esm', 'diametrvalkovmmklet8esm',

'diametrvalkovmmklet9esm',]]

zazor = dt[['zazorvalkovmmklet1mpm',

'zazorvalkovmmklet2mpm', 'zazorvalkovmmklet3mpm',

'zazorvalkovmmklet4mpm', 'zazorvalkovmmklet5mpm',

'zazorvalkovmmklet6mpm', 'zazorvalkovmmklet7mpm',

'zazorvalkovmmklet1esm', 'zazorvalkovmmklet2esm',

'zazorvalkovmmklet3esm', 'zazorvalkovmmklet4esm',

'zazorvalkovmmklet5esm', 'zazorvalkovmmklet6esm',

'zazorvalkovmmklet7esm', 'zazorvalkovmmklet8esm',

'zazorvalkovmmklet9esm']]

oborot = dt[['skorostvalkovklet1obminmpm', 'skorostvalkovklet2obminmpm',

'skorostvalkovklet3obminmpm', 'skorostvalkovklet4obminmpm',

'skorostvalkovklet5obminmpm', 'skorostvalkovklet6obminmpm',

'skorostvalkovklet7obminmpm',

'skorostkletiobminklet1esm',

'skorostkletiobminklet2esm', 'skorostkletiobminklet3esm',

'skorostkletiobminklet4esm', 'skorostkletiobminklet5esm',

'skorostkletiobminklet6esm', 'skorostkletiobminklet7esm',

'skorostkletiobminklet8esm', 'skorostkletiobminklet9esm']]

else:

#берем уникальную строку по дате + айди

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('ID'))

#берем уникальную строку по дате + айди

diam\_fact\_mpm = dt[['vtz\_report2\_29', 'vtz\_report2\_30', 'vtz\_report2\_31',

'vtz\_report2\_32', 'vtz\_report2\_33',

'vtz\_report2\_34', 'vtz\_report2\_35', 'vtz\_report2\_85',

'vtz\_report2\_86', 'vtz\_report2\_87', 'vtz\_report2\_88',

'vtz\_report2\_89', 'vtz\_report2\_90', 'vtz\_report2\_91',

'vtz\_report2\_92', 'vtz\_report2\_93']]

zazor = dt[['vtz\_report2\_43', 'vtz\_report2\_44', 'vtz\_report2\_45', 'vtz\_report2\_46',

'vtz\_report2\_47', 'vtz\_report2\_48', 'vtz\_report2\_49',

'vtz\_report2\_106', 'vtz\_report2\_107', 'vtz\_report2\_108', 'vtz\_report2\_109',

'vtz\_report2\_110', 'vtz\_report2\_111', 'vtz\_report2\_112', 'vtz\_report2\_113',

'vtz\_report2\_114']]

oborot = dt[['vtz\_report2\_36', 'vtz\_report2\_37', 'vtz\_report2\_38', 'vtz\_report2\_39',

'vtz\_report2\_40', 'vtz\_report2\_41', 'vtz\_report2\_42',

'vtz\_report2\_95', 'vtz\_report2\_96', 'vtz\_report2\_97', 'vtz\_report2\_98',

'vtz\_report2\_99', 'vtz\_report2\_100', 'vtz\_report2\_101', 'vtz\_report2\_102',

'vtz\_report2\_103']]

#округляем данные

diam\_fact\_mpm = (pd.to\_numeric(ff(diam\_fact\_mpm.values.tolist()), downcast="float", errors='coerce')).astype(float).round(1)

diam\_fact\_mpm\_new = (pd.to\_numeric(ff(diam\_fact\_mpm\_new.values.tolist()), downcast="float", errors='coerce')).astype(float).round(1)

oborot = (pd.to\_numeric(ff(oborot.values.tolist()), downcast="float", errors='coerce')).astype(float).round(0)

zazor = (pd.to\_numeric(ff(zazor.values.tolist()), downcast="float", errors='coerce')).astype(float).round(1)

skor\_zazor = return\_speed(zazor= zazor[0:7], #аналог

oborot= oborot[0:7],

Kalibr=float(Kalibr), #аналог

diam\_fact\_mpm=diam\_fact\_mpm[0:7], #аналог

diam\_fact\_mpm\_new= diam\_fact\_mpm\_new[0:7], #пользователь

data\_prop=data\_prop)

y\_data = y\_data\_float[0] #данные для графика - модель

z\_data = skor\_zazor[0] #данные для графика - аналоги

oborot\_esm = np.round((diam\_fact\_mpm[7:]\*oborot[7:])/diam\_fact\_mpm\_new[7:], 0) #пересчет оборотов для esm

tabl\_zazor = tables\_zazory(df, df\_select, skor = (skor\_zazor[1] + oborot\_esm.tolist()), zazor = zazor)

return zazor\_table(tabl\_zazor), dbc.Card(dcc.Graph(figure = graph\_2(x\_data, y\_data, z\_data),

style = {'height': "22vw", 'align': 'center'},

id = 'graph\_1'

), body = True, color="light"), {'display': 'block'}, [df\_select.to\_dict('records'),

zazor,

(skor\_zazor[1] + oborot\_esm.tolist()),

table\_data[int(row\_out[0])].get('Источник данных'),

skor\_zazor[0],

skor\_zazor[2]], {'display': 'block'}

return dash\_table.DataTable(id='zazor\_table'), no\_update, {'display': 'none'}, None, {'display': 'none'}

else:

return dash\_table.DataTable(id='zazor\_table'), no\_update, {'display': 'none'}, None, {'display': 'none'}

@app.callback(

Output("download", "data"),

[

Input("item-MPM", "n\_clicks"),

Input("item-PS", "n\_clicks"),

Input('store', 'data')

])

def update\_download\_href(n\_clicks, n\_clicks2, current\_id):

##функция, позволяющая скачивать отчеты в формате эксель. Передаем тип отчета (МПМ или ПС),

#и id, по которому подтягиваем данные из Postgres

changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

if current\_id is None:

current\_id = '0'

if 'item-MPM' in changed\_id:

df\_report = functions2.data\_by\_ID(track\_id = current\_id)

if df\_report.empty == True: #если данных нет, то возвращаем 0

series\_obj = pd.Series([0]\*df\_report.shape[1],

index=df\_report.columns)

df\_report = df\_report.append(series\_obj,

ignore\_index=True)

replace\_ = df\_report.to\_dict('list')

replace\_ = {i:str(j[0]) for i,j in replace\_.items()}

template(replace\_, card\_name ="MPM.xlsx")

return send\_file('MPM.xlsx')

elif 'item-PS' in changed\_id:

df\_report = functions2.data\_by\_ID(track\_id = current\_id)

if df\_report.empty == True: #если данных нет, то возвращаем 0

series\_obj = pd.Series([0]\*df\_report.shape[1],

index=df\_report.columns )

df\_report = df\_report.append(series\_obj,

ignore\_index=True)

replace\_ = df\_report.to\_dict('list')

replace\_ = {i:str(j[0]) for i,j in replace\_.items() }

template(replace\_, card\_name ="PS.xlsx")

return send\_file('PS.xlsx')

###calback resolve problems with filter on 2 and other pages###

@app.callback(

Output('table\_analog', "page\_current"),

Input('table\_analog', "filter\_query"))

def update\_table\_filter\_1(filter):

return 0

@app.callback(

Output('select\_table', "page\_current"),

Input('select\_table', "filter\_query"))

def update\_table\_filter\_2(filter):

return 0

###save as ideal pipes###

@app.callback(Output('load\_resultpg1\_2', 'children'), #display last update time

Output('modal\_analog\_page1', 'is\_open'),

Output('modal\_analog2\_page1', 'is\_open'),

Output('load\_resultpg1\_1', 'children'),

Input('save\_ideal\_btn\_pg1', 'n\_clicks'),

Input('closeanalogmodal\_page1', 'n\_clicks'),

Input('rejectsave\_page1', 'n\_clicks'),

Input('oksave\_page1', 'n\_clicks'),

State("table\_analog", "selected\_rows"),

State("table\_analog", "data"),

State('modal\_analog\_page1', 'is\_open'),

State('modal\_analog2\_page1', 'is\_open'),

prevent\_initial\_call=True)

def page1saveideal(btn\_save, close, reject, resave, row\_out, table\_data, is\_open, is\_open2): #(btn\_save, close, reject, save, row\_id, is\_open , is\_open2):

# print('save')

# print(btn\_save)

# print(row\_out)

old\_colnames = ['vtz\_report2\_1', 'vtz\_report2\_3', 'vtz\_report1\_1', 'vtz\_report1\_2' ,'vtz\_report1\_3',

'vtz\_report2\_29', 'vtz\_report2\_30', 'vtz\_report2\_31', 'vtz\_report2\_32', 'vtz\_report2\_33',

'vtz\_report2\_34', 'vtz\_report2\_35', 'vtz\_report2\_85', 'vtz\_report2\_86', 'vtz\_report2\_87', 'vtz\_report2\_88',

'vtz\_report2\_89', 'vtz\_report2\_90', 'vtz\_report2\_91', 'vtz\_report2\_92', 'vtz\_report2\_93',

'vtz\_report2\_43', 'vtz\_report2\_44', 'vtz\_report2\_45', 'vtz\_report2\_46','vtz\_report2\_47', 'vtz\_report2\_48', 'vtz\_report2\_49',

'vtz\_report2\_106', 'vtz\_report2\_107', 'vtz\_report2\_108', 'vtz\_report2\_109',

'vtz\_report2\_110', 'vtz\_report2\_111', 'vtz\_report2\_112', 'vtz\_report2\_113', 'vtz\_report2\_114',

'vtz\_report2\_36', 'vtz\_report2\_37', 'vtz\_report2\_38', 'vtz\_report2\_39',

'vtz\_report2\_40', 'vtz\_report2\_41', 'vtz\_report2\_42',

'vtz\_report2\_95', 'vtz\_report2\_96', 'vtz\_report2\_97', 'vtz\_report2\_98',

'vtz\_report2\_99', 'vtz\_report2\_100', 'vtz\_report2\_101', 'vtz\_report2\_102', 'vtz\_report2\_103', 'vtz\_report1\_46']

new\_colnames = ['diametrtruby', 'stenkatruby', 'kalibrelm', 'diametropravkimmmpm' ,'markastali',

'diametrvalkovmmklet1mpm','diametrvalkovmmklet2mpm', 'diametrvalkovmmklet3mpm',

'diametrvalkovmmklet4mpm', 'diametrvalkovmmklet5mpm','diametrvalkovmmklet6mpm', 'diametrvalkovmmklet7mpm',

'diametrvalkovmmklet1esm', 'diametrvalkovmmklet2esm', 'diametrvalkovmmklet3esm', 'diametrvalkovmmklet4esm',

'diametrvalkovmmklet5esm', 'diametrvalkovmmklet6esm','diametrvalkovmmklet7esm', 'diametrvalkovmmklet8esm',

'diametrvalkovmmklet9esm','zazorvalkovmmklet1mpm','zazorvalkovmmklet2mpm', 'zazorvalkovmmklet3mpm',

'zazorvalkovmmklet4mpm', 'zazorvalkovmmklet5mpm','zazorvalkovmmklet6mpm', 'zazorvalkovmmklet7mpm',

'zazorvalkovmmklet1esm', 'zazorvalkovmmklet2esm','zazorvalkovmmklet3esm', 'zazorvalkovmmklet4esm',

'zazorvalkovmmklet5esm', 'zazorvalkovmmklet6esm', 'zazorvalkovmmklet7esm', 'zazorvalkovmmklet8esm',

'zazorvalkovmmklet9esm', 'skorostvalkovklet1obminmpm', 'skorostvalkovklet2obminmpm',

'skorostvalkovklet3obminmpm', 'skorostvalkovklet4obminmpm','skorostvalkovklet5obminmpm', 'skorostvalkovklet6obminmpm',

'skorostvalkovklet7obminmpm', 'skorostkletiobminklet1esm','skorostkletiobminklet2esm', 'skorostkletiobminklet3esm',

'skorostkletiobminklet4esm', 'skorostkletiobminklet5esm','skorostkletiobminklet6esm', 'skorostkletiobminklet7esm',

'skorostkletiobminklet8esm', 'skorostkletiobminklet9esm', 'gilza']

changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

# print(changed\_id)

if 'save\_ideal\_btn\_pg1' in changed\_id:

if row\_out and row\_out != []:

text = 'check'

# print(table\_data[int(row\_out[0])].get('Источник данных'))

if table\_data[int(row\_out[0])].get('Источник данных') == 'Данные со стана':

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('Data'),

data = 'dash\_analogue', id\_ = "mpmin")

dt = dt[dt['idpipesinuque'] == table\_data[int(row\_out[0])].get('ID')]

elif table\_data[int(row\_out[0])].get('Источник данных') == 'Данные со стана (интеграция)':

#берем уникальную строку по дате + айди

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('ID'), data = 'integrazaya', id\_ = "id")

elif table\_data[int(row\_out[0])].get('Источник данных') == 'Фактические настройки': #по источнику данных определяем нужну таблицу в Postgress

#берем уникальную строку по дате + айди

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('ID'),

data = 'dash\_pipes', id\_ = "id")

else:

#берем уникальную строку по дате + айди

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('ID'))

# rename columns to save

dt.rename(columns={i:j for i,j in zip(old\_colnames,new\_colnames)}, inplace=True)

list\_for\_check = dt[['diametrtruby', 'stenkatruby', 'kalibrelm',

'diametropravkimmmpm' ,'markastali',]]

list\_check = ff(list\_for\_check.values.tolist())

list\_1 = [float(i) for i in list\_check[0:4]]

list\_1.append(list\_check[4])

####проверка на наличие####

sql\_query2 = 'WHERE diametrtruby = (%s) AND stenkatruby = (%s) AND kalibrelm = (%s) AND diametropravkimmmpm = (%s) AND markastali = (%s) ORDER BY ctid DESC LIMIT 500'

dfquery = functions2.data\_query(sql\_query2, list\_1, data = 'dash\_pipes')

if len(dfquery) > 0:

text ="Данные с диаметром: {}, стенкой: {}, калибром: {}, диаметром оправки: {} и маркой стали: {} уже существуют".format(list\_1[0],

list\_1[1], list\_1[2], list\_1[3], list\_1[4]), html.Br(), 'Пересохранить существующие данные'

return no\_update, is\_open, not is\_open2, text

columslist = functions2.sql\_data\_LIMIT500(data = 'dash\_pipes', limit = 1).columns

# if not 'gilza' in dt.columns:

# dt['gilza'] = None

dt['gilza'] = None

dt['id']=dt['diametrtruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

dt['stenkatruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

dt['diametropravkimmmpm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

dt['kalibrelm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

dt['markastali'].astype('str')

dt=dt.replace('', 0)

df\_to\_save = dt[columslist.tolist()]

sss\_heattreatment=''

for i in range(0, df\_to\_save.shape[1]):

sss\_heattreatment=sss\_heattreatment + "%s,"

sss\_heattreatment=sss\_heattreatment[:-1]

down\_status = functions2.save\_to\_sql(sql\_base = df\_to\_save,

sss\_heattreatment = sss\_heattreatment, data = "dash\_pipes",)

# print(down\_status)

text ="Данные с диаметром: {}, стенкой: {}, калибром: {}, диаметром оправки: {} и маркой стали: {} сохранены".format(ff(dt['diametrtruby']),

ff(dt['stenkatruby']), ff(dt['kalibrelm']),

ff(dt['diametropravkimmmpm']), ff(dt['markastali']),)

if down\_status == 'loaded':

text ="Данные с диаметром: {}, стенкой: {}, калибром: {}, диаметром оправки: {} и маркой стали: {} сохранены".format(ff(dt['diametrtruby']),

ff(dt['stenkatruby']), ff(dt['kalibrelm']), ff(dt['diametropravkimmmpm']), ff(dt['markastali']),)

else:

text ="Ошибка при сохранений данных. Попробуйте пересохранить еще раз"

# print(text)

return text, not is\_open, is\_open2, no\_update

else:

text = 'Выберите трубу для сохранения'

return text, not is\_open, is\_open2, no\_update

elif 'closeanalogmodal\_page1' in changed\_id:

return no\_update, not is\_open , is\_open2, no\_update

elif 'rejectsave\_page1' in changed\_id:

return no\_update, is\_open , not is\_open2, no\_update

elif 'oksave\_page1' in changed\_id:

text = 'check'

# print(table\_data[int(row\_out[0])].get('Источник данных'))

if table\_data[int(row\_out[0])].get('Источник данных') == 'Данные со стана':

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('Data'),

data = 'dash\_analogue', id\_ = "mpmin")

dt = dt[dt['idpipesinuque'] == table\_data[int(row\_out[0])].get('ID')]

elif table\_data[int(row\_out[0])].get('Источник данных') == 'Данные со стана (интеграция)':

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('ID'), data = 'integrazaya', id\_ = "id")

elif table\_data[int(row\_out[0])].get('Источник данных') == 'Фактические настройки': #по источнику данных определяем нужну таблицу в Postgress

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('ID'),

data = 'dash\_pipes', id\_ = "id")

else:

dt = functions2.data\_by\_ID(table\_data[int(row\_out[0])].get('ID'))

dt.rename(columns={i:j for i,j in zip(old\_colnames,new\_colnames)}, inplace=True)

columslist = functions2.sql\_data\_LIMIT500(data = 'dash\_pipes', limit = 1).columns

dt['gilza'] = None

dt['id']=dt['diametrtruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

dt['stenkatruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

dt['diametropravkimmmpm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

dt['kalibrelm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

dt['markastali'].astype('str')

dt=dt.replace('', 0)

df\_to\_save = dt[columslist.tolist()]

#delete duplicate###

functions2.delete\_row\_by\_ID(del\_id=ff(dt['id']), data = "dash\_pipes", id\_ = 'id')

sss\_heattreatment=''

for i in range(0, df\_to\_save.shape[1]):

sss\_heattreatment=sss\_heattreatment + "%s,"

sss\_heattreatment=sss\_heattreatment[:-1]

down\_status = functions2.save\_to\_sql(sql\_base = df\_to\_save,

sss\_heattreatment = sss\_heattreatment, data = "dash\_pipes",)

# print(down\_status)

if down\_status == 'loaded':

text ="Данные с диаметром: {}, стенкой: {}, калибром: {}, диаметром оправки: {} и маркой стали: {} пересохранены".format(ff(dt['diametrtruby']),

ff(dt['stenkatruby']), ff(dt['kalibrelm']), ff(dt['diametropravkimmmpm']), ff(dt['markastali']),)

else:

text ="Ошибка при сохранений данных. Попробуйте пересохранить еще раз"

# print(text)

return text, not is\_open, not is\_open2, no\_update

###CALBACK FROM FEDOROV FEDOR

####################################################################################################

@app.callback(

###### Output

# table\_1,table\_2,plot\_1

Output('postgres\_datatable', 'children'),

Output('postgres\_datatable2', 'children'),

Output('out-plot', 'children'),

# style for table\_1,table\_2,plot\_1

Output('postgres\_datatable', 'style'),

Output('postgres\_datatable2', 'style'),

Output('out-plot', 'style'),

# clicks to save\_to\_postgres

Output('save\_to\_postgres', 'n\_clicks'),

# clicks to del\_to\_postgres

Output('del\_to\_postgres', 'n\_clicks'),

# clicks to add rows

Output('editing-rows-button', 'n\_clicks'),

# text

Output('placeholder', 'children'),

# index

Output('index', 'n\_clicks'),

# editable tabl\_1

Output('out-table', 'editable'),

# текст кнопки add

Output('editing-rows-button', 'children'),

Output('out-table', 'selected\_rows'),

# # отмена выбора

# Output('cancel', 'n\_clicks'),

###### Input

# add rows

[Input('editing-rows-button', 'n\_clicks'),

# save\_to\_postgres

Input('save\_to\_postgres', 'n\_clicks'),

# save\_to\_postgres

Input('del\_to\_postgres', 'n\_clicks'),

# number row

# Input('out-table', 'derived\_virtual\_selected\_rows'),

Input('out-table', 'derived\_viewport\_selected\_row\_ids'),

Input("interval", "n\_intervals"),

# index

Input("index", "n\_clicks")],

###### State

# table\_1

State('out-table', 'data'),

# table\_1

State('out-table2', 'data'),

prevent\_initial\_call=True)

def df\_to\_csv\_0(add\_n\_clicks, n\_clicks,del\_clicks,row, n\_intervals, index,dataset,dataset2):

# print(add\_n\_clicks)

# print(n\_clicks)

# print(del\_clicks)

# print(row)

# print(n\_intervals)

# print(index)

# print(cancel)

# print(dataset)

# print(dataset2)

##############

pgg = pd.DataFrame(dataset)

#print(len(pgg))

# первоначальный вывод таблицы #row==[]

if add\_n\_clicks==0 and n\_clicks==0 and del\_clicks==0 and row==[] and n\_intervals==None and dataset==[] and dataset2==[]:

#print('условие 1')

return table\_1(functions.read\_sql\_ff('dash\_pipes'),row=[],data\_name=data\_name)[0],\

no\_update,\

no\_update,\

{'display': 'block'},\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

True,\

no\_update,\

no\_update

# сохранение после добавления строк

#elif add\_n\_clicks>-999999 and n\_clicks>0 and del\_clicks>-999999 and row==[] and n\_intervals==None and dataset!=[] and dataset2==[]:

elif add\_n\_clicks>-999999 and n\_clicks>0 and del\_clicks>-999999 and row==[] and n\_intervals==None and dataset!=[]:

#print('условие 2')

n\_clicks=0

pg = pd.DataFrame(dataset)

pysto=0

param=['diametropravkimmmpm','diametrtruby','stenkatruby','markastali','kalibrelm']

for i in param:

pysto+=int(pg[pg[i]==''].shape[0])

if pysto==0:

pg['id']=pg['diametrtruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['stenkatruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['diametropravkimmmpm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['kalibrelm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['markastali'].astype('str')

pg=pg.replace('', 0)

for i in pg.columns:

if i=='markastali' or i=='id':

pg[i]=pg[i].astype('object')

else:

pg[i]=pg[i].astype('float')

################# проверка на id

data\_count=pg[['id','diametropravkimmmpm','diametrtruby','stenkatruby','markastali','kalibrelm','gilza']].groupby(['id','diametropravkimmmpm','diametrtruby','stenkatruby','markastali','kalibrelm'],as\_index=False).agg('count')

data\_count.sort\_values(by='gilza',ascending=False,inplace=True)

if data\_count[data\_count['gilza']>1].shape[0]!=0:

#print('условие 2.1')

data\_count.reset\_index(drop=True,inplace=True)

to\_text=data\_count[data\_count['gilza']>1].loc[0,'id']

return table\_1(functions.read\_sql\_ff('dash\_pipes'),row=[],data\_name=data\_name)[0],\

no\_update,\

no\_update,\

{'display': 'block','editable':True},\

no\_update,\

no\_update,\

n\_clicks,\

no\_update,\

no\_update,\

html.Plaintext('данные с диаметром: '+to\_text.split('\_')[0]+', стенкой: '+to\_text.split('\_')[1]+', диаметром оправки: '+to\_text.split('\_')[2]+', калибром: '+to\_text.split('\_')[3]+', маркой стали: '+to\_text.split('\_')[4]+' уже есть в данных',style={'color': 'green', 'font-weight': 'bold', 'font-size': 'large'}),\

no\_update,\

True,\

no\_update,\

no\_update

else:

# print('условие 2.2')

# print(len(pg))

#functions.update\_sql\_ff(pg,'dash\_pipes')

functions.update\_sql\_ff(pg.drop\_duplicates(keep='last'),'dash\_pipes')

return table\_1(functions.read\_sql\_ff('dash\_pipes'),row=[],data\_name=data\_name)[0],\

no\_update,\

no\_update,\

{'display': 'block','editable':True},\

no\_update,\

no\_update,\

n\_clicks,\

no\_update,\

no\_update,\

html.Plaintext("данные сохранены "+str(datetime.now() + timedelta(hours=3)),style={'color': 'green', 'font-weight': 'bold', 'font-size': 'large'}),\

no\_update,\

True,\

no\_update,\

no\_update

else:

return no\_update,\

no\_update,\

no\_update,\

{'display': 'block','editable':True},\

no\_update,\

no\_update,\

n\_clicks,\

no\_update,\

no\_update,\

html.Plaintext('заполните все поля',style={'color': 'green', 'font-weight': 'bold', 'font-size': 'large'}),\

no\_update,\

True,\

no\_update,\

no\_update

# выбoр по активной строке

elif add\_n\_clicks>-999999 and n\_clicks==0 and del\_clicks>-999999 and row!=[] and n\_intervals==None and dataset!=[] and dataset2!=-999999 and list(pgg[pgg['id']==row[0]].index)[0]!=index:

#print('условие 3')

# index=row[0]

pg = pd.DataFrame(dataset)

#print(pg)

index=list(pg[pg['id']==row[0]].index)[0]

pysto=0

param=['diametropravkimmmpm','diametrtruby','stenkatruby','markastali','kalibrelm']

for i in param:

pysto+=int(pg[pg[i]==''].shape[0])

if pysto==0:

pg['id']=pg['diametrtruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['stenkatruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['diametropravkimmmpm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['kalibrelm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['markastali'].astype('str')

pg=pg.replace('', 0)

for i in pg.columns:

if i=='markastali' or i=='id':

pg[i]=pg[i].astype('object')

else:

pg[i]=pg[i].astype('float')

################# проверка на id

data\_count=pg[['id','diametropravkimmmpm','diametrtruby','stenkatruby','markastali','kalibrelm','gilza']].groupby(['id','diametropravkimmmpm','diametrtruby','stenkatruby','markastali','kalibrelm'],as\_index=False).agg('count')

data\_count.sort\_values(by='gilza',ascending=False,inplace=True)

if data\_count[data\_count['gilza']>1].shape[0]!=0:

data\_count.reset\_index(drop=True,inplace=True)

to\_text=data\_count[data\_count['gilza']>1].loc[0,'id']

return no\_update,\

no\_update,\

no\_update,\

{'display': 'block'},\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

html.Plaintext('данные с диаметром: '+to\_text.split('\_')[0]+', стенкой: '+to\_text.split('\_')[1]+', диаметром оправки: '+to\_text.split('\_')[2]+', калибром: '+to\_text.split('\_')[3]+', маркой стали: '+to\_text.split('\_')[4]+' уже есть в данных',style={'color': 'green', 'font-weight': 'bold', 'font-size': 'large'}),\

index,\

False,\

'Отмена выбранной строки',\

no\_update

else:

table\_2\_plot\_2=table\_1(pd.DataFrame(dataset).replace('', 0),selected\_rows=list(pg[pg['id']==row[0]].index),row=list(pg[pg['id']==row[0]].index),data\_name=data\_name)

try:

fedorovfedor=pd.DataFrame(table\_2\_plot\_2[1])

return no\_update,\

table\_2\_plot\_2[1],\

table\_2\_plot\_2[2],\

{'display': 'block'},\

{'display': 'block'},\

{'display': 'block'},\

no\_update,\

no\_update,\

no\_update,\

[],\

index,\

False,\

'Отмена выбранной строки',\

no\_update

except:

return table\_1(pd.DataFrame(dataset),row=[],data\_name=data\_name)[0],\

no\_update,\

no\_update,\

{'display': 'block'},\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

html.Plaintext('Сначала сохраните данные',style={'color': 'red', 'font-weight': 'bold', 'font-size': 'large'}),\

index,\

False,\

no\_update,\

no\_update

else:

return no\_update,\

no\_update,\

no\_update,\

{'display': 'block'},\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

html.Plaintext('сначала заполните все пустые поля',style={'color': 'green', 'font-weight': 'bold', 'font-size': 'large'}),\

index,\

False,\

'Отмена выбранной строки',\

no\_update

# добавление строки

# elif add\_n\_clicks>0 and add\_n\_clicks!=-999999 and n\_clicks>-999999 and del\_clicks>-999999\

# and row==[] and n\_intervals==None and dataset!=[] and dataset2==[]:

elif add\_n\_clicks>0 and add\_n\_clicks!=-999999 and n\_clicks>-999999 and del\_clicks>-999999 and row==[] and n\_intervals==None and dataset!=[]:

#print('условие 4')

add\_n\_clicks=0

pg = pd.DataFrame(dataset)

pysto=0

param=['diametropravkimmmpm','diametrtruby','stenkatruby','markastali','kalibrelm']

for i in param:

pysto+=int(pg[pg[i]==''].shape[0])

if pysto==0:

pg['id']=pg['diametrtruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['stenkatruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['diametropravkimmmpm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['kalibrelm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

pg['markastali'].astype('str')

pg=pg.replace('', 0)

for i in pg.columns:

if i=='markastali' or i=='id':

pg[i]=pg[i].astype('object')

else:

pg[i]=pg[i].astype('float')

data\_count=pg[['id','diametropravkimmmpm','diametrtruby','stenkatruby','markastali','kalibrelm','gilza']].groupby(['id','diametropravkimmmpm','diametrtruby','stenkatruby','markastali','kalibrelm'],as\_index=False).agg('count')

data\_count.sort\_values(by='gilza',ascending=False,inplace=True)

if data\_count[data\_count['gilza']>1].shape[0]!=0:

data\_count.reset\_index(drop=True,inplace=True)

to\_text=data\_count[data\_count['gilza']>1].loc[0,'id']

return no\_update,\

no\_update,\

no\_update,\

{'display': 'block'},\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

add\_n\_clicks,\

html.Plaintext('данные с диаметром: '+to\_text.split('\_')[0]+', стенкой: '+to\_text.split('\_')[1]+', диаметром оправки: '+to\_text.split('\_')[2]+', калибром: '+to\_text.split('\_')[3]+', маркой стали: '+to\_text.split('\_')[4]+' уже есть в данных',style={'color': 'green', 'font-weight': 'bold', 'font-size': 'large'}),\

no\_update,\

True,\

no\_update,\

no\_update

else:

dataset.append(dict(zip(list(data\_name['name\_bd']),[''] \* len(list(data\_name['name\_bd'])))))

dataset=[dataset[-1]]+dataset[:-1]

return table\_1(pd.DataFrame(dataset),row=[],data\_name=data\_name)[0],\

no\_update,\

no\_update,\

{'display': 'block'},\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

add\_n\_clicks,\

no\_update,\

no\_update,\

True,\

no\_update,\

no\_update

else:

return no\_update,\

no\_update,\

no\_update,\

{'display': 'block'},\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

add\_n\_clicks,\

html.Plaintext('сначала заполните все пустые поля',style={'color': 'green', 'font-weight': 'bold', 'font-size': 'large'}),\

no\_update,\

True,\

no\_update,\

no\_update

# отмена выбора

elif add\_n\_clicks>0 and n\_clicks>-999999 and del\_clicks>-999999 and row!=[] and n\_intervals==None and dataset!=[] and dataset2!=-999999:

#print('условие 5')

add\_n\_clicks=0

index=-1

return no\_update,\

no\_update,\

no\_update,\

{'display': 'block'},\

{'display': 'none'},\

{'display': 'none'},\

no\_update,\

no\_update,\

add\_n\_clicks,\

[],\

index,\

True,\

'Добавить',\

[]

# сохранение 2-х табличек

elif add\_n\_clicks>-999999 and n\_clicks>0 and del\_clicks>-999999 and row!=[] and n\_intervals==None and dataset!=[] and dataset2!=[]:

#elif add\_n\_clicks>-999999 and n\_clicks>0 and del\_clicks>-999999 and n\_intervals==None and dataset!=[] and dataset2!=[]:

if row[0] == '':

#print('ok')

dataset2\_1 = [

{'стан': 'МПМ', 'Номер клети': 1, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'МПМ', 'Номер клети': 2, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'МПМ', 'Номер клети': 3, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'МПМ', 'Номер клети': 4, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'МПМ', 'Номер клети': 5, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'МПМ', 'Номер клети': 6, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'МПМ', 'Номер клети': 7, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'ИКС', 'Номер клети': 1, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'ИКС', 'Номер клети': 2, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'ИКС', 'Номер клети': 3, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'ИКС', 'Номер клети': 4, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'ИКС', 'Номер клети': 5, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'ИКС', 'Номер клети': 6, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'ИКС', 'Номер клети': 7, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'ИКС', 'Номер клети': 8, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0},

{'стан': 'ИКС', 'Номер клети': 9, 'Диаметр Валков': 0, 'Обороты': 0, 'Зазоры': 0}]

else:

dataset2\_1 = dataset2

#print('условие 6')

n\_clicks=0

text = df\_to\_csv\_2(list(pgg[pgg['id']==row[0]].index),dataset,dataset2\_1)

# print(text)

table\_2\_plot\_2=table\_1(functions.read\_sql\_ff('dash\_pipes'),selected\_rows=list(pgg[pgg['id']==row[0]].index),row=list(pgg[pgg['id']==row[0]].index),data\_name=data\_name)

return table\_2\_plot\_2[0],\

table\_2\_plot\_2[1],\

table\_2\_plot\_2[2],\

no\_update,\

no\_update,\

no\_update,\

n\_clicks,\

no\_update,\

no\_update,\

text,\

no\_update,\

False,\

no\_update,\

no\_update

# таблица 1

# таблица 2

# график 1

# видимость табл 1

# видимость табл 2

# видимость графика 1

# кол-во нажатий на сохранить

# кол-во нажатий на del

# кол-во нажатий на add rows

# текст

# невидимая кнопка с выбором строки

# редактирование табл 1

# текст кнапки add

# отмена выбранной строки

# удаление

elif add\_n\_clicks>-999999 and n\_clicks>-999999 and del\_clicks>0 and row!=[] and n\_intervals==None and dataset!=[] and dataset2!=[]:

del\_clicks=0

pg = pd.DataFrame(dataset)

# print(ff(pg['id'][list(pg[pg['id']==row[0]].index)]))

functions2.delete\_row\_by\_ID(del\_id=ff(pg['id'][list(pg[pg['id']==row[0]].index)]), data = "dash\_pipes", id\_ = 'id')

# print('условие 7 - удаление')

index = -1

return table\_1(functions.read\_sql\_ff('dash\_pipes'),row=[],data\_name=data\_name)[0],\

no\_update,\

no\_update,\

no\_update,\

{'display': 'none'},\

{'display': 'none'},\

no\_update,\

del\_clicks,\

no\_update,\

[],\

index,\

no\_update,\

'Добавить',\

[]

elif add\_n\_clicks>-999999 and n\_clicks>-999999 and del\_clicks>0 and row!=[] and n\_intervals==None and dataset!=[] and dataset2==[]:

del\_clicks=0

pg = pd.DataFrame(dataset)

functions2.delete\_row\_by\_ID(del\_id=ff(pg['id'][list(pg[pg['id']==row[0]].index)]), data = "dash\_pipes", id\_ = 'id')

# print('условие 7.2 - удаление')

index = -1

return table\_1(functions.read\_sql\_ff('dash\_pipes'),row=[],data\_name=data\_name)[0],\

no\_update,\

no\_update,\

no\_update,\

{'display': 'none'},\

{'display': 'none'},\

no\_update,\

del\_clicks,\

no\_update,\

[],\

index,\

no\_update,\

'Добавить',\

[]

else:

# print('else')

return no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update,\

no\_update

####################################################################################################

###calback resolve problems with filter on 2 and other pages###

@app.callback(

Output('out-table', "page\_current"),

Input('out-table', "filter\_query"))

def update\_table\_filter\_3(filter):

return 0

@app.callback(

Output('real\_pipes', "page\_current"),

Input('real\_pipes', "filter\_query"))

def update\_table\_filter\_4(filter):

#print(filter)

return 0

@app.callback(Output('updatetime', 'children'), #display last update time

Output('page3\_maintable', 'children'), #display default table

Output('real\_pipes', 'selected\_rows'), #unselect selected row

Output('modal\_page3', 'is\_open'), #modal for chosing pipes by condition

Output('selector\_kalibrelm', 'options'), #unique options for dropdown

Output('selector\_markastali', 'options'), #unique options for dropdown

Output('selector\_diametrtruby', 'options'), #unique options for dropdown

Output('selector\_stenkatruby', 'options'), #unique options for dropdown

Output('selector\_diametropravkimmmpm', 'options'), #unique options for dropdown

#graphs which emerged when selected row

Output('page3\_table', 'children'), #table1

Output('page3\_graph', 'children'), #graph speed

Output('toki\_mpm', 'children'), #graph current

Output('toki\_esm', 'children'), #graph current

Input('renew\_data', 'n\_clicks'),

Input('condition\_load\_btn', 'n\_clicks'),

Input('ok\_page3', 'n\_clicks'),

Input('return\_page3', 'n\_clicks'),

Input('selector\_kalibrelm', 'value'),

Input('selector\_markastali', 'value'),

Input('selector\_diametrtruby', 'value'),

Input('selector\_stenkatruby', 'value'),

Input('selector\_diametropravkimmmpm', 'value'),

Input('date\_from', 'value'),

Input('date\_to', 'value'),

#Input('real\_pipes', 'selected\_rows'),

Input('real\_pipes', 'derived\_viewport\_selected\_row\_ids'),

State('modal\_page3', 'is\_open'),

prevent\_initial\_call=True)

def page3update(n, selectdata, ok, refuse, kalibrelm\_val, markastali\_val, diametrtruby\_val, stenkatruby\_val, diametropravkimmmpm\_val, date\_from, date\_to, row\_id, is\_open):

changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

query\_date\_to = (datetime.now() + timedelta(hours=3)).strftime("%Y-%m-%d %H:%M:%S")

query\_date\_from = '1990-01-00 00:00:00'

if 'renew\_data' in changed\_id:

return html.P('Обновлено: ' + str((datetime.now() + timedelta(hours=3)).strftime("%d-%m-%Y %H:%M:%S"))), table\_realpipes(functions2.sql\_data\_LIMIT500(data = 'integrazaya', order\_id = 'timestamp', limit = 10), 0)[0], [], no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, None, None, None, None

elif 'condition\_load\_btn' in changed\_id:

#upload unique variables for dropdowns from Postgress

check = functions2.sql\_data\_select(vals = 'kalibrelm', data = 'integrazaya')

kalibrelm = [float(ff(n)) for n in check]

check = functions2.sql\_data\_select(vals = 'diametropravkimmmpm', data = 'integrazaya')

diametropravkimmmpm = [float(ff(n)) for n in check]

check = functions2.sql\_data\_select(vals = 'diametrtruby', data = 'integrazaya')

diametrtruby = [float(ff(n)) for n in check]

check = functions2.sql\_data\_select(vals = 'stenkatruby', data = 'integrazaya')

stenkatruby = [float(ff(n)) for n in check]

check = functions2.sql\_data\_select(vals = 'markastali', data = 'integrazaya')

markastali = [ff(n) for n in check]

return no\_update, no\_update, no\_update, not is\_open, [{'label': i, 'value': i} for i in kalibrelm], [{'label': i, 'value': i} for i in markastali], [{'label': i, 'value': i} for i in diametrtruby], [{'label': i, 'value': i} for i in stenkatruby], [{'label': i, 'value': i} for i in diametropravkimmmpm], no\_update, no\_update, no\_update, no\_update

elif 'return\_page3' in changed\_id: #undo query for pipes

return no\_update, no\_update, no\_update, not is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update

elif 'ok\_page3' in changed\_id: #pipes query by condition

#create query

query\_PSQ = query\_kalibrelm = query\_diametrtruby = query\_stenkatruby = query\_diametropravkimmmpm = query\_markastali =''

sql\_list = []

if kalibrelm\_val and len(kalibrelm\_val) > 0:

query\_kalibrelm = 'kalibrelm = ANY (%s) AND '

sql\_list.append(kalibrelm\_val)

if diametrtruby\_val and len(diametrtruby\_val) > 0:

query\_diametrtruby = 'diametrtruby = ANY (%s) AND '

sql\_list.append(diametrtruby\_val)

if stenkatruby\_val and len(stenkatruby\_val) > 0:

query\_stenkatruby = 'stenkatruby = ANY (%s) AND '

sql\_list.append(stenkatruby\_val)

if diametropravkimmmpm\_val and len(diametropravkimmmpm\_val) > 0:

query\_diametropravkimmmpm = 'diametropravkimmmpm = ANY (%s) AND '

sql\_list.append(diametropravkimmmpm\_val)

if markastali\_val and len(markastali\_val) > 0:

query\_markastali = 'markastali = ANY (%s) AND '

sql\_list.append(markastali\_val)

if date\_from and date\_from != []:

query\_date\_from = datetime.fromisoformat(date\_from).strftime("%Y-%m-%d %H:%M:%S")

sql\_list.append(query\_date\_from)

if date\_to and date\_to != []:

#delta = timedelta.Timedelta(hours=23, minutes = 59, seconds = 59)

delta = timedelta(hours=23, minutes = 59, seconds = 59)

query\_date\_to = (datetime.fromisoformat(date\_to) + delta).strftime("%Y-%m-%d %H:%M:%S")

sql\_list.append(query\_date\_to)

query\_PSQ = 'WHERE ' + query\_kalibrelm + query\_diametrtruby + query\_stenkatruby + query\_diametropravkimmmpm + query\_markastali

query\_PSQ = query\_PSQ + 'timestamp BETWEEN (%s) and (%s) ORDER BY timestamp DESC LIMIT 500'

return html.P('Обновлено: ' + str((datetime.now() + timedelta(hours=3)).strftime("%d-%m-%Y %H:%M:%S"))), table\_realpipes(functions2.data\_query(query\_PSQ, sql\_list), 0)[0], [], not is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, None, None, None, None

elif 'real\_pipes' in changed\_id:

if row\_id and row\_id != []:

# print(row\_id)

currents = functions2.data\_by\_ID(track\_id = ff(row\_id), data = 'current', id\_ = "id") #currents

df\_by\_id = functions2.data\_by\_ID(track\_id = ff(row\_id), data = 'integrazaya', id\_ = "id") #rows data

graph\_1 = graph\_2 = None

if currents\_graph(currents)[0]:

graph\_1 = dcc.Graph(figure = currents\_graph(currents)[0], style = {'height': '400px'})

if currents\_graph(currents)[1]:

graph\_2 = dcc.Graph(figure = currents\_graph(currents)[1], style = {'height': '400px'})

return no\_update, no\_update, no\_update, is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, table\_realpipes(df\_by\_id, 0)[1], table\_realpipes(df\_by\_id, 0)[2], graph\_1, graph\_2

return no\_update, no\_update, no\_update, is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update

else:

return no\_update, no\_update, no\_update, is\_open, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update, no\_update

###save as ideal pipes###

@app.callback(Output('load\_result2', 'children'), #display last update time

Output('modal\_analog\_page3', 'is\_open'),

Output('modal\_analog2\_page3', 'is\_open'),

Output('load\_result', 'children'),

Input('save\_ideal\_btn', 'n\_clicks'),

Input('closeanalogmodal\_page3', 'n\_clicks'),

Input('rejectsave\_page3', 'n\_clicks'),

Input('oksave\_page3', 'n\_clicks'),

State('real\_pipes', 'derived\_viewport\_selected\_row\_ids'),

State('modal\_analog\_page3', 'is\_open'),

State('modal\_analog2\_page3', 'is\_open'),

prevent\_initial\_call=True)

def page3saveideal(btn\_save, close, reject, save, row\_id, is\_open , is\_open2):

# print('save')

# print(row\_id)

changed\_id = [p['prop\_id'] for p in dash.callback\_context.triggered][0]

if 'save\_ideal\_btn' in changed\_id:

if row\_id and row\_id != []:

df\_by\_id = functions2.data\_by\_ID(track\_id = ff(row\_id),

data = 'integrazaya', id\_ = "id")[['diametrtruby', 'stenkatruby', 'kalibrelm',

'diametropravkimmmpm' ,'markastali']]

list\_check = ff(df\_by\_id.values.tolist())

list\_1 = [float(i) for i in list\_check[0:4]]

list\_1.append(list\_check[4])

####проверка на наличие####

sql\_query2 = 'WHERE diametrtruby = (%s) AND stenkatruby = (%s) AND kalibrelm = (%s) AND diametropravkimmmpm = (%s) AND markastali = (%s) ORDER BY ctid DESC LIMIT 500'

dfquery = functions2.data\_query(sql\_query2, list\_1, data = 'dash\_pipes')

if len(dfquery) > 0:

text ="Данные с диаметром: {}, стенкой: {}, калибром: {}, диаметром оправки: {} и маркой стали: {} уже существуют".format(list\_1[0],

list\_1[1], list\_1[2], list\_1[3], list\_1[4]), html.Br(), 'Пересохранить существующие данные'

return no\_update, is\_open , not is\_open2, text

else:

df\_by\_id = functions2.data\_by\_ID(track\_id = ff(row\_id),

data = 'integrazaya', id\_ = "id")

df\_by\_id['id']=df\_by\_id['diametrtruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

df\_by\_id['stenkatruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

df\_by\_id['diametropravkimmmpm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

df\_by\_id['kalibrelm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

df\_by\_id['markastali'].astype('str')

df\_by\_id=df\_by\_id.replace('', 0)

#functions2.delete\_row\_by\_ID(del\_id=ff(df\_by\_id['id']), data = "dash\_pipes", id\_ = 'id')

columslist = functions2.sql\_data\_LIMIT500(data = 'dash\_pipes', limit = 1).columns

df\_by\_id['gilza'] = None

df\_to\_save = df\_by\_id[columslist.tolist()]

sss\_heattreatment=''

for i in range(0, df\_to\_save.shape[1]):

sss\_heattreatment=sss\_heattreatment + "%s,"

sss\_heattreatment=sss\_heattreatment[:-1]

down\_status = functions2.save\_to\_sql(sql\_base = df\_to\_save,

sss\_heattreatment = sss\_heattreatment, data = "dash\_pipes",)

# print(down\_status)

if down\_status == 'loaded':

text ="Данные с диаметром: {}, стенкой: {}, калибром: {}, диаметром оправки: {} и маркой стали: {} сохранены".format(ff(df\_by\_id['diametrtruby']),

ff(df\_by\_id['stenkatruby']), ff(df\_by\_id['kalibrelm']), ff(df\_by\_id['diametropravkimmmpm']), ff(df\_by\_id['markastali']),)

else:

text ="Ошибка при сохранений данных. Попробуйте пересохранить еще раз1"

# print(text)

return text, not is\_open , is\_open2, no\_update

# return no\_update, is\_open , not is\_open2

else:

return 'Выберите трубу для сохранения', not is\_open , is\_open2, no\_update

elif 'oksave\_page3' in changed\_id:

df\_by\_id = functions2.data\_by\_ID(track\_id = ff(row\_id),

data = 'integrazaya', id\_ = "id")

df\_by\_id['id']=df\_by\_id['diametrtruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

df\_by\_id['stenkatruby'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

df\_by\_id['diametropravkimmmpm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

df\_by\_id['kalibrelm'].apply(lambda x: format(float(x), '.2f'))+'\_'+\

df\_by\_id['markastali'].astype('str')

df\_by\_id=df\_by\_id.replace('', 0)

functions2.delete\_row\_by\_ID(del\_id=ff(df\_by\_id['id']), data = "dash\_pipes", id\_ = 'id')

columslist = functions2.sql\_data\_LIMIT500(data = 'dash\_pipes', limit = 1).columns

df\_by\_id['gilza'] = None

df\_to\_save = df\_by\_id[columslist.tolist()]

sss\_heattreatment=''

for i in range(0, df\_to\_save.shape[1]):

sss\_heattreatment=sss\_heattreatment + "%s,"

sss\_heattreatment=sss\_heattreatment[:-1]

down\_status = functions2.save\_to\_sql(sql\_base = df\_to\_save,

sss\_heattreatment = sss\_heattreatment, data = "dash\_pipes",)

# print(down\_status)

text ="Данные с диаметром: {}, стенкой: {}, калибром: {}, диаметром оправки: {} и маркой стали: {} пересохранены".format(ff(df\_by\_id['diametrtruby']),

ff(df\_by\_id['stenkatruby']), ff(df\_by\_id['kalibrelm']),

ff(df\_by\_id['diametropravkimmmpm']), ff(df\_by\_id['markastali']),)

if down\_status == 'loaded':

text ="Данные с диаметром: {}, стенкой: {}, калибром: {}, диаметром оправки: {} и маркой стали: {} сохранены".format(ff(df\_by\_id['diametrtruby']),

ff(df\_by\_id['stenkatruby']), ff(df\_by\_id['kalibrelm']), ff(df\_by\_id['diametropravkimmmpm']), ff(df\_by\_id['markastali']),)

else:

text ="Ошибка при сохранений данных. Попробуйте пересохранить еще раз"

return text, not is\_open , not is\_open2, no\_update

elif 'closeanalogmodal\_page3' in changed\_id:

return no\_update, not is\_open , is\_open2, no\_update

elif 'rejectsave\_page3' in changed\_id:

return no\_update, is\_open , not is\_open2, no\_update

# Update the index

@app.callback(Output('page-content', 'children'),

[Input('url', 'pathname')])

def display\_page(pathname):

if pathname == '/':

return page\_1\_layout

# elif pathname == '/model':

# return page\_1\_layout

elif pathname == '/table':

return page\_2\_layout

elif pathname == '/realpipes':

page\_3\_layout = dbc.Container([

dcc.Interval(id='interval\_pg3', interval=60000, n\_intervals=0),# interval update - 15 minites # interval=60000\*15

html.P(),

# dbc.Row([dbc.Col(group3, width = 10),

# dbc.Col(html.P('Обновлено: ' + str((datetime.now() + timedelta(hours=3)).strftime("%d-%m-%Y %H:%M:%S")), style = {'font-size': '10'}),

# width = 2, id = 'updatetime')]),

dbc.Row([dbc.Col(group3, width = 12)]),

dbc.Row([dbc.Col(html.P('Обновлено: ' + str((datetime.now() + timedelta(hours=3)).strftime("%d-%m-%Y %H:%M:%S")), style = {'font-size': '10'}),

width = 12, id = 'updatetime')]),

# html.P(),

dbc.Row(

[

dbc.Col(

html.Div(table\_realpipes(functions2.sql\_data\_LIMIT500(data = 'integrazaya', order\_id = 'timestamp', limit = 10), 0)[0], id = 'page3\_maintable'),

width = 12)]),

html.P(),

dbc.Row(

[

dbc.Col(html.Div(id = 'page3\_table'), width = 5),

dbc.Col(html.Div(id = 'page3\_graph'), width = 7),

]),

dbc.Row([

dbc.Col(html.Div(id = 'toki\_mpm'), width = 6),

dbc.Col(html.Div(id = 'toki\_esm'), width = 6)

]),

],fluid = True)

return page\_3\_layout

else:

return page\_1\_layout

if \_\_name\_\_ == '\_\_main\_\_':

app.run\_server(debug=False, host = '0.0.0.0', port = 8050)