In [ ]:

```
import plotly.io as pio
import plotly.express as px

pio.renderers.default = "notebook"
```

For these calculations we use several datasets downloaded from Bureau of Transportation Statistics (transtats.bts.gov):

- domestic (USA) air transportation statistics for July, years: 2014, 2019 and 2023 (including number of passengers and cargo transported);
- international air transportation statistics for July, years: 2014, 2019 and 2023 (including number of passengers and cargo transported); E.g. there are 6 (six) data sets (cvs files) used.

Here we calculate % change in number of passengers or cargo pounds moved by airlines between two time poins spaced 5 years apart. We used two time periods: pre-COVID years pair (July of 2014-July of 2019) and COVID + after-COVID years pair (July of 2019 - July of 2023). The first year in pair is labeled as year "x" and the later year is labelled as year "y".

These caulations can be used for any other years or months user finds interesting to look into.

These calculations will allow to answer a question whether flow of passengers and cargo via airports or countries might changed after COVID in comparison to pre-covid time-period (assuming that the rest of the monthes that year have similar passenger and cargo flows).

For USA domestic market we calculated % of change on the level of airports and for international air transportation market we calculated % of cnang on the level of countries.

Calculations scheme is:

- group rows in a cvs file on origin-destination and calculate sums of passengers or cargo pounds transported;

- merge resulted tables (dataframes) for a given pair of years using ('outer' merge). This will allow us to detect lost or acquired destinations later.

- for USA domestic market only, we removed all pairs of origin-destination having less than 1000 passengers transported per month (July). This is done because there are multiple aircraft relocation flights or unplanned flights which can not be considered regular routes. Therefore we assume that if airline transports more than 1000 passengers per month per route, that will be a commercial route. We are only interested in commercial routes.

- Then label each destination between years x and y as acquired, lost or no_change, using conditions: n passengers equal or more than zero etc in year x versus the same for year y. If n passenders or cargo dissapears in year y, this destination is labelled as lost; if passengers or cargo canged from zero to above zero in year y - destination is acquired; no_change label is given when there n passengers and cargo is above zero.

Having destiantions labeled we group rows by origins and calculate how many destinations are lost or acquired or have no_change. The list of destinations is recorded for each category.

% of change is calculates for an origin as delta in all passengers (or all cargo) transported for years x and y divided by n passengers (or cargo) for year x. For origins destination pairs where the only destination was acquired, % cnage is 100%; for the only one lost destination, % change is -100%.

The results are plotted on three figures: two scatterplots and one bar plot. For 1st scatterplot we only plot origins having more than 100K passengers per month. For barplot and 2d scatterplot, we only plot selected largest airports.

I wanted to clreate a universal file for calculations so that at the beginning a user indicates files s/he wants to use, years (used in the files names), cargo or passengers and then the calculations take place automatically and created cvs files are stored on the disk (writing result files on disk is disabled by #.

In [ ]:

```
import pandas as pd
import numpy as np
import random
import string
import matplotlib.pyplot as plt
import plotly.express as px

airports=['AFW', 'ANC', 'ATL', 'ATL', 'CVG', 'DFW', 'EWR', 'HNL', 'IAH', 'IND', 'INL', 'JFK', 'LAX',
          'MIA', 'MEM', 'OAK', 'ONT', 'ORD', 'PDX', 'PHL','PHX','RFD', 'SBD', 'SDF', 'SEA']

countries=['MX', 'CA', 'GB', 'FR', 'DE', 'NL', 'JP', 'CN', 'BR', 'IT', 'ES', 'KR']
```

In [ ]:

```
x = input("\n Enter first year: 2014, 2019 or 2023: ")
y = input("\n Enter last year: 2014, 2019 or 2023: ")
type1 = input("\n Type dom or intl" )
type2 = input("\n Type pax or cargo")

if type1 == "dom":
  type3 = 'airpt'
else:
  type3 = 'countr'

if type1 == "dom":
  selection = airports
else:
  selection = countries
```

In [ ]:

```
file_name_x = type1 + "_july_" + x + ".csv"
file_name_y = type1 + "_july_" + y + ".csv"

print("\n file_name_x:", file_name_x)
print("\n type", type(file_name_x))
print("\n file_name_y:", file_name_y)

# load data
df_20x_i=pd.read_csv(file_name_x)
df_20y_i=pd.read_csv(file_name_y)
```

```
file_name_x: dom_july_2014.csv

type <class 'str'>

file_name_y: dom_july_2019.csv
```

In [ ]:

```
# names will be used later to store temp dataframes
df_20x_i.name='y20x'
df_20y_i.name='y20y'

#creating list of dataframes. Note with 'df' will be stored as string
df_names=[df_20x_i, df_20y_i] # without '' can whole df acan be called

# checking df names
for i in df_names:
    print(i.name)

# adding new column cargo
for i in df_names:
    i['CARGO']=i['FREIGHT']+i['MAIL']

# checking column names in one of the loaded df
df_20x_i.keys()
```

```
y20x
y20y
```

Out[ ]:

```
Index(['DEPARTURES_SCHEDULED', 'DEPARTURES_PERFORMED', 'PAYLOAD', 'SEATS',
       'PASSENGERS', 'FREIGHT', 'MAIL', 'DISTANCE', 'RAMP_TO_RAMP', 'AIR_TIME',
       'UNIQUE_CARRIER', 'AIRLINE_ID', 'UNIQUE_CARRIER_NAME',
       'UNIQUE_CARRIER_ENTITY', 'REGION', 'CARRIER', 'CARRIER_NAME',
       'CARRIER_GROUP', 'CARRIER_GROUP_NEW', 'ORIGIN_AIRPORT_ID',
       'ORIGIN_AIRPORT_SEQ_ID', 'ORIGIN_CITY_MARKET_ID', 'ORIGIN',
       'ORIGIN_CITY_NAME', 'ORIGIN_STATE_ABR', 'ORIGIN_STATE_FIPS',
       'ORIGIN_STATE_NM', 'ORIGIN_WAC', 'DEST_AIRPORT_ID',
       'DEST_AIRPORT_SEQ_ID', 'DEST_CITY_MARKET_ID', 'DEST', 'DEST_CITY_NAME',
       'DEST_STATE_ABR', 'DEST_STATE_FIPS', 'DEST_STATE_NM', 'DEST_WAC',
       'AIRCRAFT_GROUP', 'AIRCRAFT_TYPE', 'AIRCRAFT_CONFIG', 'YEAR', 'QUARTER',
       'MONTH', 'DISTANCE_GROUP', 'CLASS', 'CARGO'],
      dtype='object')
```

In [ ]:

```
if type1 == "dom":
  no=df_20x_i.columns.get_loc("ORIGIN")
  nd=df_20x_i.columns.get_loc("DEST")
else:
  no=df_20x_i.columns.get_loc("ORIGIN_COUNTRY")
  nd=df_20x_i.columns.get_loc("DEST_COUNTRY")

if type2 == "pax":
  np=df_20x_i.columns.get_loc("PASSENGERS")
else:
  np=df_20x_i.columns.get_loc("CARGO")

print('\n index, np =', np, ', column name:', df_20x_i.columns[np], \
'\n index, no =', no, ', column name:', df_20x_i.columns[no], \
'\n index, nd =', nd, ', column name:', df_20x_i.columns[nd])

print("\n type1:", type1, "\n type2:", type2, "\n type3:", type3)

file_label= type1 + "_by_" + type3 + "_" + type2 + "_yy" +x +"_"+y

print("\n file_label", file_label)
```

```
index, np = 45 , column name: CARGO
index, no = 22 , column name: ORIGIN
index, nd = 31 , column name: DEST

type1: dom
type2: cargo
type3: airpt

file_label dom_by_airpt_cargo_yy2014_2019
```

In [ ]:

```
# group each df for 20x_i and 20y_i by O (origin) and D (destination), sum number of passengers
# creating column OD
#
# item_sum in the code below is the name of the new column (item is pax or cargo)
# col - column; oper - operator
# def - creates function

def gr_df(df, col, oper):
    result = df.groupby(
    [i.columns[no],i.columns[nd]]
    ).agg(
        item_sum=(col, oper)).reset_index()

    return result

for i in df_names:
    # file name to record df
    name=i.name
    print(name)
    # group by
    result = gr_df(i, i.columns[np], 'sum')
    print(name, 'after_gr', result.keys())
    # new OD column
    result['OD'] = result[[i.columns[no], i.columns[nd]]].agg('-'.join, axis=1)
    print(name, '+OD', result.keys())
    result1=result.copy()
    if type2 == "pax":
        # leave only rows with npas >1000
        result1= result[result['item_sum'] > 1000]
        print(name, 'rows >1000', result.keys())

    # SAVING temporary dataframe as csv file with name
    result1.to_csv("temp_"+name +'.csv')
```

```
y20x
y20x after_gr Index(['ORIGIN', 'DEST', 'item_sum'], dtype='object')
y20x +OD Index(['ORIGIN', 'DEST', 'item_sum', 'OD'], dtype='object')
y20y
y20y after_gr Index(['ORIGIN', 'DEST', 'item_sum'], dtype='object')
y20y +OD Index(['ORIGIN', 'DEST', 'item_sum', 'OD'], dtype='object')
```

In [ ]:

```
# calculating variables

# load two created temporary files
y20x=pd.read_csv('temp_y20x.csv', index_col=0)
y20y=pd.read_csv('temp_y20y.csv', index_col=0)

# merge two y20x and y20y dataframes to find out what OD appeared, lost or got no change
merged_yx_yy=y20x.merge(y20y, how='outer', on='OD')

import numpy as np

def calculations(df):

    df_copy=df.copy()

    # classification task
    # replace nan with zeroes
    df_copy.fillna(0, inplace=True)
    # create additional columns O and D
    df_copy[['O', 'D']] = df_copy['OD'].str.split('-', n=1, expand=True)

    # classification conditions
    conditions = [
    (df_copy['item_sum_x'] == 0) & (df_copy['item_sum_y'] > 0),
    (df_copy['item_sum_x'] >0 ) & (df_copy['item_sum_y'] == 0),
    (df_copy['item_sum_x'] >0 ) & (df_copy['item_sum_y'] > 0)
    ]

    # create a list of the values to assign for each condition:
    # acquired = a, lost = l, no_change= n
    values = ['a', 'l', 'n']
    # applying conditions
    df_copy['outcome'] = np.select(conditions, values)

    # code to save any intermediate df. For example,
    # df_copy.to_csv(file_label+"outcome"+".csv")

    # excuding unnedded columns
    df_copy=df_copy[['item_sum_x', 'item_sum_y', 'OD', 'O', 'D', 'outcome']]

    # for each origin (aport) calculate # destinations a, l , n.
    df_copy=df_copy.groupby(
    ['O','outcome'], as_index=False).agg(
    count=('O', 'count'),
    # record destinations as a list
    list_D=('D', lambda x: list(x)),
    # calc sums of passengers
    item_tot_x=('item_sum_x', 'sum'),
    item_tot_y=('item_sum_y', 'sum')
    )

    # SAVING FILE
    # df_copy.to_csv(file_label+ "_interm_df" + ".csv")
    # instead of saving file, create intermediate_df
    intermediate_df=df_copy.copy()

    # change columns format (dtype) for downstream operations
    df_copy['srt_count']=df_copy['count'].astype(str)

    df_copy['list_D']=df_copy['list_D'].astype(str)
    df_copy['O']=df_copy['O'].astype(str)

    # creating columns for records of # of a, l, n and lists of aports
    df_copy['status'] = df_copy[['outcome', 'srt_count']].agg('='.join, axis=1)
    df_copy['list_D_upd'] = df_copy[['outcome', 'list_D']].agg('='.join, axis=1)
    df_copy['notes'] = df_copy[['status', 'list_D_upd']].agg(' ; '.join, axis=1)

    # group by by origin
    df_copy=df_copy.groupby('O').agg(item_abstot_x=('item_tot_x', 'sum'),
                               item_abstot_y=('item_tot_y', 'sum'),
                               summary_outcome=('status', lambda x: list(x)),
                             list_apts=('list_D_upd', lambda x: list(x)),
                               ).reset_index()

    df_copy['summary_outcome']=df_copy['summary_outcome'].astype(str)
    df_copy['list_apts']=df_copy['list_apts'].astype(str)

    # calculate absolute change in pass numbers
    df_copy['item_y_x']=df_copy['item_abstot_y']-df_copy['item_abstot_x']

    # calc % change
    df_copy['%_change'] = np.where(df_copy['item_abstot_x'] == 0, 100, round(100*df_copy['item_y_x']/df_copy['item_abstot_x'], 2))
    # delta in pax for years x and y
    df_copy['item_y_x']=df_copy['item_abstot_y']-df_copy['item_abstot_x']

    result=df_copy

    return result

calculated_df=calculations(merged_yx_yy)
# merged_yx_yy_cl merged file for two years x and y

# SAVING FILE
# calculated_df.to_csv(file_label + "calcul_df" + ".csv")

calculated_df.head(5)
```

Out[ ]:

	O	item_abstot_x	item_abstot_y	summary_outcome	list_apts	item_y_x	%_change
0	05A	958.0	1313.0	['0=1', 'a=2', 'l=2']	["0=['AET']", "a=['FAI', 'AKP']", "l=['ARC', '...	355.0	37.06
1	08A	300.0	0.0	['l=1']	["l=['HOM']"]	-300.0	-100.00
2	09A	0.0	0.0	['0=2']	["0=['ADQ', 'ORI']"]	0.0	100.00
3	1B1	0.0	0.0	['0=2']	["0=['ACK', 'LEW']"]	0.0	100.00
4	1G4	0.0	0.0	['0=6']	["0=['BLD', 'GCN', 'KNB', 'VGT', '1G4', 'DQS']"]	0.0	100.00

In [ ]:

```
# file_name=file_label+ "_interm_df" + ".csv"
# intermediate_df=pd.read_csv(file_name, index_col=0)
```

In [ ]:

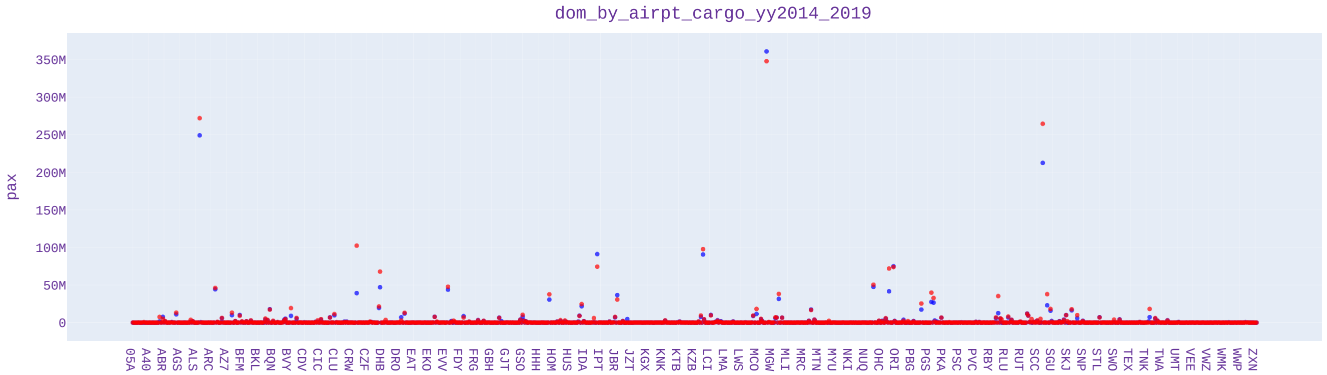
```
def for_plot (df):

    df_copy=df.copy()
    # recording sign on % change
    df_copy['sign'] = (np.where(df_copy['%_change'] < 0, "- ", "+ ")).astype(str)

    # recording only numbers in % change
    df_copy['%_change_str'] = (abs(df_copy['%_change'])).astype(str)

   # new column for data labels (% change as str)
    df_copy['%_change_str']=df_copy['sign'] + df_copy['%_change_str'] + "%"

    if type2 == "pax":
        # filtering df to have only rows with item >100 000
        result =df_copy[(df_copy['item_abstot_x']>100000) |(df_copy['item_abstot_y']>100000) ]

    # splitting df_copy into 2 df for year x and year y
    # for year x, skipping % change and list_apts
    df_copy_x=df_copy[['O', 'item_abstot_x', 'summary_outcome']].copy()
    # adding year columns
    df_copy_x['year']=x
    # changing column name to cargo
    df_copy_x.rename(columns={"item_abstot_x": "item"}, inplace=True)

    # for year y, skipping summary_outcome and NOT skipping % change and list_apts
    df_copy_y=df_copy[['O', 'item_abstot_y', 'list_apts', '%_change_str']].copy()
    # adding year columns
    df_copy_y['year']=y
    # changing column name to cargo
    df_copy_y.rename(columns={"item_abstot_y": "item"}, inplace=True)

    # concatenate
    result = pd.concat([df_copy_x, df_copy_y], axis=0)


    return result

df_4plot=for_plot(calculated_df)

# SAVING FILE
# df_4plot.to_csv(file_label + "plotting" + ".csv")

df_4plot.head(5)
```

Out[ ]:

	O	item	summary_outcome	year	list_apts	%_change_str
0	05A	958.0	['0=1', 'a=2', 'l=2']	2014	NaN	NaN
1	08A	300.0	['l=1']	2014	NaN	NaN
2	09A	0.0	['0=2']	2014	NaN	NaN
3	1B1	0.0	['0=2']	2014	NaN	NaN
4	1G4	0.0	['0=6']	2014	NaN	NaN

In [ ]:

```
df_4plot_1=for_plot(calculated_df)

fig1 = px.scatter(df_4plot_1, x="O", y="item", color="year", color_discrete_sequence=["blue", "red"],
                 opacity=0.7,
                 hover_data=["summary_outcome", "%_change_str"])

fig1.update_layout(title={'text': file_label,
                          'y':0.95,
                          'x':0.5,
           'xanchor': 'center',
           'yanchor': 'top'},
             xaxis_title="Airports",
             yaxis_title="pax",
             legend_title="Years",
             font=dict(
        family="Courier New, monospace",
        size=16,
        color="RebeccaPurple"
    )
)
# fig1.update_yaxes(range=[0, 5000000])
fig1.show()

# fig1.write_html("scatter_1" + file_label + ".html")
```



In [ ]:

```
calculated_df.describe()
```

Out[ ]:

	item_abstot_x	item_abstot_y	item_y_x	%_change
count	1.010000e+03	1.010000e+03	1.010000e+03	1010.000000
mean	2.107655e+06	2.428919e+06	3.212635e+05	378.533455
std	1.657988e+07	1.789278e+07	3.261158e+06	7974.916764
min	0.000000e+00	0.000000e+00	-1.671457e+07	-100.000000
25%	0.000000e+00	0.000000e+00	-4.375000e+03	-36.537500
50%	2.026500e+03	1.330500e+03	0.000000e+00	19.835000
75%	3.889000e+04	3.379500e+04	6.025000e+01	100.000000
max	3.609257e+08	3.478952e+08	6.316686e+07	250864.710000

In [ ]:

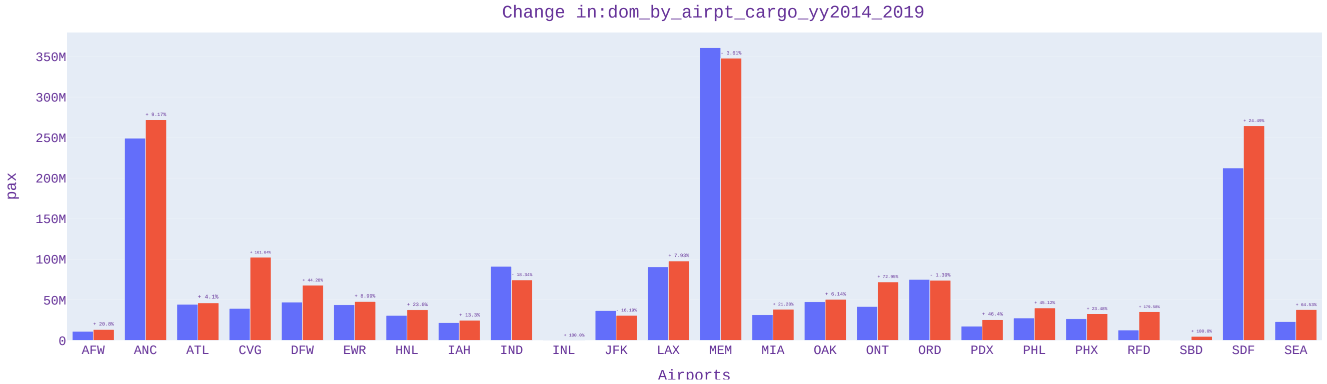
```
df_largest_aports=calculated_df.loc[calculated_df['O'].isin(selection)] #selected origins
df_largest_aports.head(5)
```

Out[ ]:

	O	item_abstot_x	item_abstot_y	summary_outcome	list_apts	item_y_x	%_change
39	AFW	10967725.0	13248948.0	['0=1', 'a=6', 'l=2', 'n=15']	["0=['DFW']", "a=['LFT', 'MCI', 'MSP', 'SJC', ...	2281223.0	20.80
60	ANC	249187916.0	272040711.0	['0=26', 'a=15', 'l=18', 'n=66']	["0=['AIN', 'AUK', 'KLG', 'KLN', 'KPV', 'NIN',...	22852795.0	9.17
74	ATL	44314245.0	46129989.0	['0=53', 'a=14', 'l=10', 'n=113']	["0=['ABY', 'ACY', 'AEX', 'AMA', 'AZO', 'BMI',...	1815744.0	4.10
201	CVG	39223221.0	102390083.0	['0=48', 'a=13', 'l=7', 'n=38']	["0=['ACY', 'BGR', 'COS', 'DAB', 'DAY', 'GRR',...	63166862.0	161.04
222	DFW	47027236.0	67850160.0	['0=49', 'a=29', 'l=17', 'n=113']	["0=['ACY', 'ALB', 'AVP', 'CRW', 'CVN', 'FWA',...	20822924.0	44.28

In [ ]:

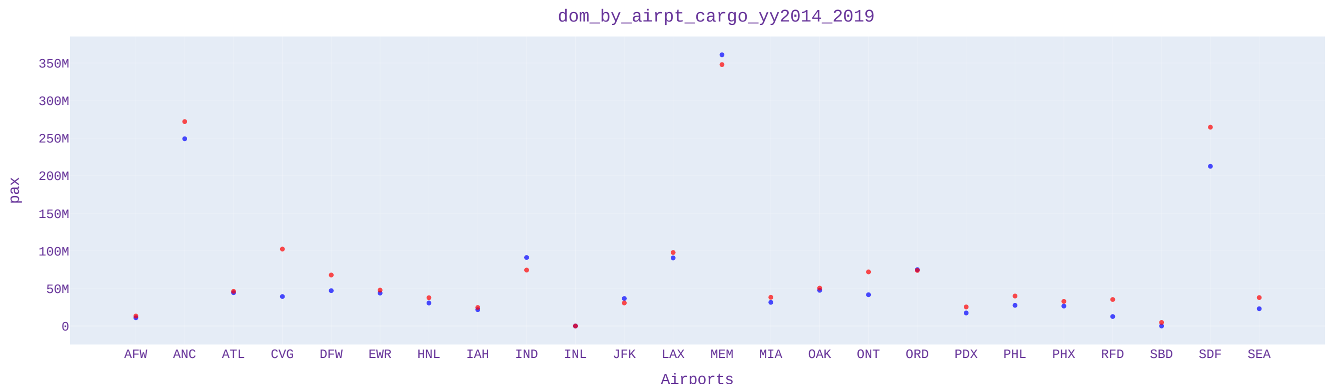
```
# selected largest aports/countries

df_4plot_2=for_plot(df_largest_aports)

fig2 = px.bar(df_4plot_2, x="O", y="item", color='year', text='%_change_str', hover_data=["summary_outcome"], barmode="group")

fig2.update_layout(title={'text': "Change in:"+file_label,
                          'y':0.95,
                          'x':0.5,
            'xanchor': 'center',
            'yanchor': 'top'},
              xaxis_title="Airports",
              yaxis_title="pax",
              legend_title="Years",
              font=dict(
        family="Courier New, monospace",
        size=16,
        color="RebeccaPurple"
    )
)
# fig2.update_yaxes(range=[0, 2000000])

fig2.update_traces(textfont_size=25, textangle=0, textposition="outside", cliponaxis=False)

fig2.show()

# fig2.write_html("bar_" + file_label + ".html")
```



In [ ]:

```
fig1 = px.scatter(df_4plot_2, x="O", y="item", color="year", color_discrete_sequence=["blue", "red"],
                 opacity=0.7,
                 hover_data=["summary_outcome", "%_change_str"])

fig1.update_layout(title={'text': file_label,
                          'y':0.95,
                          'x':0.5,
            'xanchor': 'center',
            'yanchor': 'top'},
              xaxis_title="Airports",
              yaxis_title="pax",
              legend_title="Years",
              font=dict(
        family="Courier New, monospace",
        size=16,
        color="RebeccaPurple"
```

```
    )
)
# fig1.update_yaxes(range=[0, 5000000])
fig1.show()

# fig1.write_html("scatter_2" + file_label + ".html")
```



In [ ]:
```
print(calculated_df.describe())
df_4plot
```

```
       item_abstot_x  item_abstot_y      item_y_x       %_change
count   1.010000e+03   1.010000e+03  1.010000e+03    1010.000000
mean    2.107655e+06   2.428919e+06  3.212635e+05     378.533455
std     1.657988e+07   1.789278e+07  3.261158e+06    7974.916764
min     0.000000e+00   0.000000e+00 -1.671457e+07    -100.000000
25%     0.000000e+00   0.000000e+00 -4.375000e+03     -36.537500
50%     2.026500e+03   1.330500e+03  0.000000e+00      19.835000
75%     3.889000e+04   3.379500e+04  6.025000e+01     100.000000
max     3.609257e+08   3.478952e+08  6.316686e+07  250864.710000
```

Out[ ]:

	O	item	summary_outcome	year	list_apts	%_change_str
0	05A	958.0	['0=1', 'a=2', 'l=2']	2014	NaN	NaN
1	08A	300.0	['l=1']	2014	NaN	NaN
2	09A	0.0	['0=2']	2014	NaN	NaN
3	1B1	0.0	['0=2']	2014	NaN	NaN
4	1G4	0.0	['0=6']	2014	NaN	NaN
...	...	...	...	...	...	...
1005	ZXB	1906.0	NaN	2019	["a=['ABQ']"]	+ 100.0%
1006	ZXH	423.0	NaN	2019	["l=['DOF', 'HYG', 'WFB']", "n=['HYL']"]	- 67.11%
1007	ZXM	0.0	NaN	2019	["0=['CGA', 'HYL', 'ZXN']", "l=['KTB', 'KTN', ...	- 100.0%
1008	ZXN	43.0	NaN	2019	["0=['CGA', 'HYL', 'KTB', 'WFB', 'ZXM']", "a=[...	+ 100.0%
1009	ZXU	0.0	NaN	2019	["0=['ACK', 'TN8']"]	+ 100.0%

2020 rows × 6 columns

In [ ]:
```
pivoted_1 = intermediate_df.pivot(index="O", columns="outcome", values="count").reset_index() # .rename_axis(None, axis=1)
pivoted_1=pivoted_1.loc[pivoted_1['O'].isin(airports)].reset_index(drop=True) #selected aports
pivoted_1[['a', 'l', 'n']]= pivoted_1[['a', 'l', 'n']].replace('', np.nan).astype('Int64')
pivoted=pivoted_1.merge(df_largest_aports, on='O', how='outer')
# temp=pivoted[['O', 'a', 'l', 'n', 'cargo_abstot_x', 'cargo_abstot_y', '%_change']]
# pivoted[['O', 'a', 'l', 'n', 'item_abstot_x', 'item_abstot_y', '%_change']].to_csv(file_label+ "pivot" + ".csv")
print(file_label)
pivoted.head(5)
```

```
dom_by_airpt_cargo_yy2014_2019
```

Out[ ]:

	O	0	a	l	n	item_abstot_x	item_abstot_y	summary_outcome	list_apts	item_y_x	%_change
0	AFW	1.0	6	2	15	10967725.0	13248948.0	['0=1', 'a=6', 'l=2', 'n=15']	["0=['DFW']", "a=['LFT', 'MCI', 'MSP', 'SJC', ...	2281223.0	20.80
1	ANC	26.0	15	18	66	249187916.0	272040711.0	['0=26', 'a=15', 'l=18', 'n=66']	["0=['AIN', 'AUK', 'KLG', 'KLN', 'KPV', 'NIN',...	22852795.0	9.17
2	ATL	53.0	14	10	113	44314245.0	46129989.0	['0=53', 'a=14', 'l=10', 'n=113']	["0=['ABY', 'ACY', 'AEX', 'AMA', 'AZO', 'BMI',...	1815744.0	4.10
3	CVG	48.0	13	7	38	39223221.0	102390083.0	['0=48', 'a=13', 'l=7', 'n=38']	["0=['ACY', 'BGR', 'COS', 'DAB', 'DAY', 'GRR',...	63166862.0	161.04
4	DFW	49.0	29	17	113	47027236.0	67850160.0	['0=49', 'a=29', 'l=17', 'n=113']	["0=['ACY', 'ALB', 'AVP', 'CRW', 'CVN', 'FWA',...	20822924.0	44.28

In [ ]:
```
''' Initially calculations were cheched on synthetic data.
# code in this cell is to create synthetic data
# empty dataframe
temp_df=pd.DataFrame()

# dataframe with origins and destinations,n passengers and column with random zeroes

# column O - random letters from first to 13th in alphabet
temp_df['O'] = random.choices(string.ascii_letters[0:13], k=1000)
# column D - random letters from 13th to last (26th() in alphabet
temp_df['D'] = random.choices(string.ascii_letters[14:26], k=1000)
# column with random numbers from 1 to 10000
temp_df['n_pas19'] = np.random.randint(1, 10000, size=1000)
# column with random numbers from 1 to 10000
temp_df['n_pas23'] = np.random.randint(1, 10000, size=1000)
# column with random numbers from 0 to 1
temp_df['Ran_zero1'] = temp_df.apply(lambda _: random.randint(0, 1), axis=1)
# column with random numbers from 0 to 1
temp_df['Ran_zero2'] = temp_df.apply(lambda _: random.randint(0, 1), axis=1)
# deletings randolmly rows (multiplyig by zero or 1)
temp_df['n_pas19'] = temp_df['n_pas19']* temp_df['Ran_zero1']
# deletings randolmly rows (multiplyig by zero or 1)
temp_df['n_pas23'] = temp_df['n_pas23']* temp_df['Ran_zero2']
print('\n temp_df',temp_df.keys())
# creating synthetic data for year 2019
df_2019=temp_df[['O', 'D', 'n_pas19']].copy()
# deleting rows with zero passangers (product of multiplication by zero)
df_2019= df_2019[df_2019['n_pas19'] != 0]

# creating synthetic data for year 2023
df_2023=temp_df[['O', 'D', 'n_pas23']].copy()
# deleting rows with zero passangers (product of multiplication by zero)
df_2023= df_2023[df_2023['n_pas23'] != 0]
print('\n df_2019',df_2019.keys())
print('\n df_2023',df_2023.keys())'''
```

Out[ ]:
```
" Initially calculations were cheched on synthetic data.\n# code in this cell is to create synthetic data\n# empty dataframe\ntemp_df=pd.DataFrame()\n\n# dataframe with origins and destinations,n passengers and co
lumn with random zeroes\n\n# column O - random letters from first to 13th in alphabet\ntemp_df['O'] = random.choices(string.ascii_letters[0:13], k=1000)\n# column D - random letters from 13th to last (26th() in al
phabet\ntemp_df['D'] = random.choices(string.ascii_letters[14:26], k=1000)\n# column with random numbers from 1 to 10000\ntemp_df['n_pas19'] = np.random.randint(1, 10000, size=1000)\n# column with random numbers f
rom 1 to 10000\ntemp_df['n_pas23'] = np.random.randint(1, 10000, size=1000)\n# column with random numbers from 0 to 1\ntemp_df['Ran_zero1'] = temp_df.apply(lambda _: random.randint(0, 1), axis=1)\n# column with ra
ndom numbers from 0 to 1\ntemp_df['Ran_zero2'] = temp_df.apply(lambda _: random.randint(0, 1), axis=1)\n# deletings randolmly rows (multiplyig by zero or 1)\ntemp_df['n_pas19'] = temp_df['n_pas19']* temp_df['Ran_z
ero1']\n# deletings randolmly rows (multiplyig by zero or 1)\ntemp_df['n_pas23'] = temp_df['n_pas23']* temp_df['Ran_zero2']\nprint('\n temp_df',temp_df.keys())\n# creating synthetic data for year 2019\ndf_2019=tem
p_df[['O', 'D', 'n_pas19']].copy()\n# deleting rows with zero passangers (product of multiplication by zero)\ndf_2019= df_2019[df_2019['n_pas19'] != 0]\n\n# creating synthetic data for year 2023\ndf_2023=temp_df
[['O', 'D', 'n_pas23']].copy()\n# deleting rows with zero passangers (product of multiplication by zero)\ndf_2023= df_2023[df_2023['n_pas23'] != 0]\nprint('\n df_2019',df_2019.keys())\nprint('\n df_2023',df_2023.k
eys())"
```