In [1]:
```
import re
import os
import json
import datetime as dt
import numpy as np
import pandas as pd

import seaborn as sns
import seaborn.objects as so
import matplotlib.pyplot as plt

from interact_graph import init_plot
```

In [2]:
```
key_group = ['title', 'vendor', 'country']
INPUT_DF = 'processed_df.parquet'
```

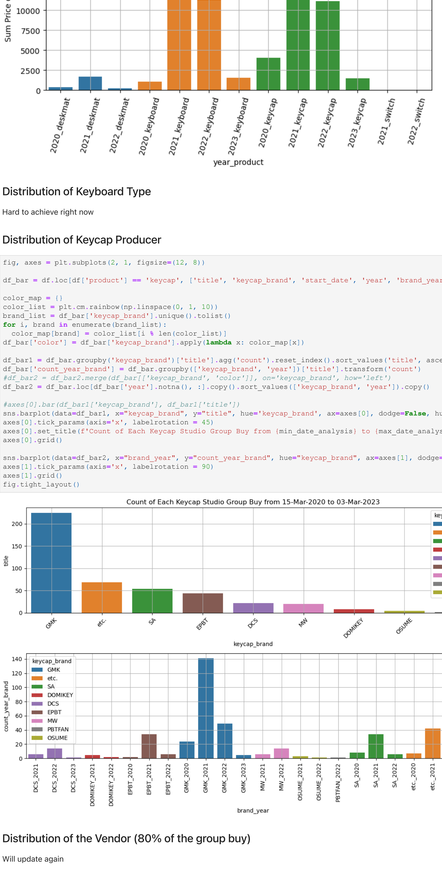
In [3]:
```
df = pd.read_parquet(INPUT_DF)

df['year'] = df['start_date'].dt.year
cond = df['year'].notna()

df['year_product'] = df.loc[cond, 'year'].astype(int).astype(str)  + "_" + df.loc[cond, 'product'].astype(str)
df['product_year'] = df.loc[cond, 'product'].astype(str) + "_" + df.loc[cond, 'year'].astype(int).astype(str)

# Keycap brand
df.loc[cond, 'year_brand'] = df.loc[cond, 'year'].astype(int).astype(str)  + "_" + df.loc[cond, 'keycap_brand']
df.loc[cond, 'brand_year'] = df.loc[cond, 'keycap_brand'].astype(str) + "_" + df.loc[cond, 'year'].astype(int).a

min_date_analysis = df['start_date'].min().strftime('%d-%b-%Y')
max_date_analysis = df['start_date'].max().strftime('%d-%b-%Y')
```

In [4]:
```
df.shape , df[df['base_price'].isna()].shape, df[df['start_date'].isna()].shape, df[df['end_date'].isna()].shape
```

Out[4]:
```
((20509, 23), (1046, 23), (721, 23), (0, 23))
```

Distribution of price

In [5]:
```
fig, axes = plt.subplots(2, 1, figsize=(8,10))

df_bar = df.loc[(df['base_price'].notna()) & (df['product'].notna()), :].copy().drop_duplicates(['title', 'prod
df_bar['base_price'] = df_bar['base_price'].astype(float)
df_bar['sum_base_price'] = df_bar.groupby('product_year')['base_price'].transform('sum')
df_bar['count_base_price'] = df_bar.groupby('product_year')['base_price'].transform('count')
df_bar = df_bar.sort_values('product_year')

sns.barplot(data=df_bar, x="product_year", y="count_base_price", hue="product", ax=axes[0], dodge=False)
axes[0].tick_params(axis='x', labelrotation = 75)
axes[0].set_title(f'Count of each Mech Product (USD) from {min_date_analysis} to {max_date_analysis}')
axes[0].set_ylabel(f'Product Count')
axes[0].grid()
axes[0].legend()

sns.barplot(data=df_bar, x="year_product", y="sum_base_price", hue="product", ax=axes[1], dodge=False)
axes[1].tick_params(axis='x', labelrotation = 75)
axes[1].set_title(f'Market Cap of each Mech Product (USD) from {min_date_analysis} to {max_date_analysis}')
axes[1].set_ylabel(f'Sum Price of Product')
axes[1].grid()
axes[1].legend()
fig.tight_layout()
```

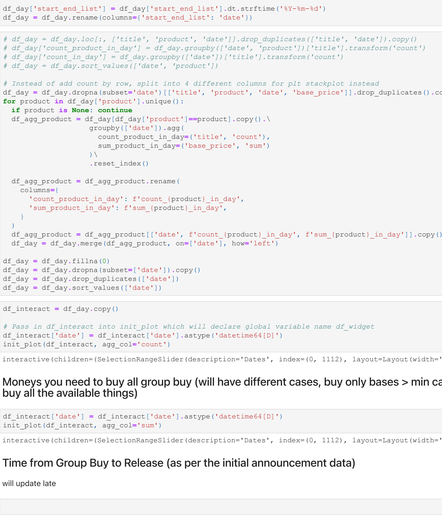


Distribution of Keyboard Type

Hard to achieve right now

Distribution of Keycap Producer

In [6]:
```
fig, axes = plt.subplots(2, 1, figsize=(12, 8))

df_bar = df.loc[df['product'] == 'keycap', ['title', 'keycap_brand', 'start_date', 'year', 'brand_year']].copy(

color_map = {}
color_list = plt.cm.rainbow(np.linspace(0, 1, 10))
brand_list = df_bar['keycap_brand'].unique().tolist()
for i, brand in enumerate(brand_list):
  color_map[brand] = color_list[i % len(color_list)]
df_bar['color'] = df_bar['keycap_brand'].apply(lambda x: color_map[x])

df_bar1 = df_bar.groupby('keycap_brand')['title'].agg('count').reset_index().sort_values('title', ascending=Fal
df_bar['count_year_brand'] = df_bar.groupby(['keycap_brand', 'year'])['title'].transform('count')
#df_bar2 = df_bar2.merge(df_bar[['keycap_brand', 'color']], on='keycap_brand', how='left')
df_bar2 = df_bar.loc[df_bar['year'].notna(), :].copy().sort_values(['keycap_brand', 'year']).copy()

#axes[0].bar(df_bar1['keycap_brand'], df_bar1['title'])
sns.barplot(data=df_bar1, x="keycap_brand", y="title", hue='keycap_brand', ax=axes[0], dodge=False, hue_order=b
axes[0].tick_params(axis='x', labelrotation = 45)
axes[0].set_title(f'Count of Each Keycap Studio Group Buy from {min_date_analysis} to {max_date_analysis}')
axes[0].grid()

sns.barplot(data=df_bar2, x="brand_year", y="count_year_brand", hue="keycap_brand", ax=axes[1], dodge=False, hu
axes[1].tick_params(axis='x', labelrotation = 90)
axes[1].grid()
fig.tight_layout()
```



Distribution of the Vendor (80% of the group buy)

Will update again

In [7]:
```
# fig, ax = plt.subplots(figsize=(10, 4))
# df_bar = df.loc[:, ['vendor', 'title']].copy()
# df_bar = df_bar.groupby('vendor')['title'].agg('count').reset_index().sort_values('title', ascending=False)
# df_bar['cumsum'] = df_bar['title'].cumsum()
# df_bar['percentile'] = df_bar['cumsum'] / df_bar['title'].sum()
# df_bar = df_bar.loc[df_bar['percentile'] > 0.8, :].copy()

# df_bar = df_bar.reset_index().iloc[:20]

# #df_bar
# ax.bar(df_bar['vendor'], df_bar['title'])
# ax.tick_params(axis='x', labelrotation = 90)
```

Visualization of number of active group buy in a Year (maybe a stack chart of available group buys/ price need to buy everything, etc.)

Would need a lot of data transformation to be in daily level (the goal is to have. a representation in each day, how many groupbuy are available, what are trey, etc.)

In [8]:
```
df_day = df[df['start_date'].notna() & (df['end_date']!='sold out')].copy()
df_day['end_date'] = df_day['end_date'].astype('datetime64[D]')
df_day['start_end_list'] = df_day.apply(lambda x: [x['start_date'] + dt.timedelta(days=i) for i in range((x['en
df_day = df_day.explode('start_end_list')

df_day['start_end_list'] = df_day['start_end_list'].dt.strftime('%Y-%m-%d')
df_day = df_day.rename(columns={'start_end_list': 'date'})
```

In [9]:
```
# df_day = df_day.loc[:, ['title', 'product', 'date']].drop_duplicates(['title', 'date']).copy()
# df_day['count_product_in_day'] = df_day.groupby(['date', 'product'])['title'].transform('count')
# df_day['count_in_day'] = df_day.groupby(['date'])['title'].transform('count')
# df_day = df_day.sort_values(['date', 'product'])

# Instead of add count by row, split into 4 different columns for plt stackplot instead
df_day = df_day.dropna(subset='date')[['title', 'product', 'date', 'base_price']].drop_duplicates().copy()
for product in df_day['product'].unique():
  if product is None: continue
  df_agg_product = df_day[df_day['product']==product].copy().\
                    groupby(['date']).agg(
                      count_product_in_day=('title', 'count'),
                      sum_product_in_day=('base_price', 'sum')
                    ).\
                    reset_index()

  df_agg_product = df_agg_product.rename(
    columns={
      'count_product_in_day': f'count_{product}_in_day',
      'sum_product_in_day': f'sum_{product}_in_day',
    }
  )
  df_agg_product = df_agg_product[['date', f'count_{product}_in_day', f'sum_{product}_in_day']].copy()
  df_day = df_day.merge(df_agg_product, on='date', how='left')

df_day = df_day.fillna(0)
df_day = df_day.dropna(subset='date').copy()
df_day = df_day.drop_duplicates(['date'])
df_day = df_day.sort_values(['date'])
```

In [10]:
```
df_interact = df_day.copy()

# Pass in df_interact into init_plot which will declare global variable name df_widget
df_interact['date'] = df_interact['date'].astype('datetime64[D]')
init_plot(df_interact, agg_col='count')

interactive(children=(SelectionRangeSlider(description='Dates', index=(0, 1112), layout=Layout(width='800px'),…
```

Moneys you need to buy all group buy (will have different cases, buy only bases > min cases, buy all the available things)

In [11]:
```
df_interact['date'] = df_interact['date'].astype('datetime64[D]')
init_plot(df_interact, agg_col='sum')

interactive(children=(SelectionRangeSlider(description='Dates', index=(0, 1112), layout=Layout(width='800px'),…
```

Time from Group Buy to Release (as per the initial announcement data)

will update late

In [ ]:
```
```