retail-analysis

July 12, 2024

1 Sales Analysis

Import libraries

[1]:
```
import pandas as pd
import os
import matplotlib.pyplot as plt
```

Merge each month's data into one csv

[2]:
```
path = r"C:\Sales Data\SalesAnalysis\Sales_Data"
files = [file for file in os.listdir("C:\Sales Data\SalesAnalysis\Sales_Data")]

all_months_data = pd.DataFrame()

for file in files:
    file_path = os.path.join(path, file)  # Use os.path.join to construct file␣
 ↪path
    current_data = pd.read_csv(file_path)
    all_months_data = pd.concat([all_months_data, current_data])

all_months_data.to_csv("all_data_copy.csv", index=False)
```

Read in updated DataFrame

[3]:
```
all_data = pd.read_csv("all_data.csv")
all_data.head()
```

[3]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
1	NaN	NaN	NaN	NaN	NaN	NaN
2	176559	Bose SoundSport Headphones	1	99.99		
3	176560	Google Phone	1	600		
4	176560	Wired Headphones	1	11.99		

```
2  04/07/19 22:30     682 Chestnut St, Boston, MA 02215
3  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
4  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
```

Clean up the data

Drop rows of NaN

[4]:
```
nan_df = all_data[all_data.isna().any(axis=1)]
nan_df.head()

all_data = all_data.dropna(how = 'all')
all_data.head()
```

[4]:
```
  Order ID                    Product Quantity Ordered Price Each  \
0   176558        USB-C Charging Cable                2      11.95
2   176559  Bose SoundSport Headphones                1      99.99
3   176560                Google Phone                1        600
4   176560            Wired Headphones                1      11.99
5   176561            Wired Headphones                1      11.99

       Order Date                      Purchase Address
0  04/19/19 08:46          917 1st St, Dallas, TX 75001
2  04/07/19 22:30     682 Chestnut St, Boston, MA 02215
3  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
4  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
5  04/30/19 09:27     333 8th St, Los Angeles, CA 90001
```

Find OR and delete it

[5]:
```
all_data = all_data[all_data['Order Date'].str[0:2] != 'Or']
all_data.head()
```

[5]:
```
  Order ID                    Product Quantity Ordered Price Each  \
0   176558        USB-C Charging Cable                2      11.95
2   176559  Bose SoundSport Headphones                1      99.99
3   176560                Google Phone                1        600
4   176560            Wired Headphones                1      11.99
5   176561            Wired Headphones                1      11.99

       Order Date                      Purchase Address
0  04/19/19 08:46          917 1st St, Dallas, TX 75001
2  04/07/19 22:30     682 Chestnut St, Boston, MA 02215
3  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
4  04/12/19 14:38  669 Spruce St, Los Angeles, CA 90001
5  04/30/19 09:27     333 8th St, Los Angeles, CA 90001
```

Convert columns to the correct type

Data Exploration

[6]:
```
all_data['Quantity Ordered'] = pd.to_numeric(all_data['Quantity Ordered']) #␣
↪make int
all_data['Price Each'] = pd.to_numeric(all_data['Price Each']) # make float
all_data.head()
```

[6]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001

Augument data with additional columns

Add Month column

[7]:
```
all_data['Month'] = all_data['Order Date'].str[0:2].astype('int32')
all_data.head()
```

[7]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4

Add a City column

[8]:
```
# Let's use .apply()

def get_city(address):
    return address.split(',')[1]
```

```
def get_state(address):
    return address.split(',')[2].split(' ')[1]

all_data['City'] = all_data['Purchase Address'].apply(lambda x: f"{get_city(x)}␣
 ↪({get_state(x)})")
all_data.head()
```

[8]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	Dallas (TX)
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	Boston (MA)
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles (CA)
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles (CA)
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	Los Angeles (CA)

Add a Sales column

```
all_data['Sales'] = all_data['Quantity Ordered'] * all_data['Price Each']
all_data.head()
```

[9]:

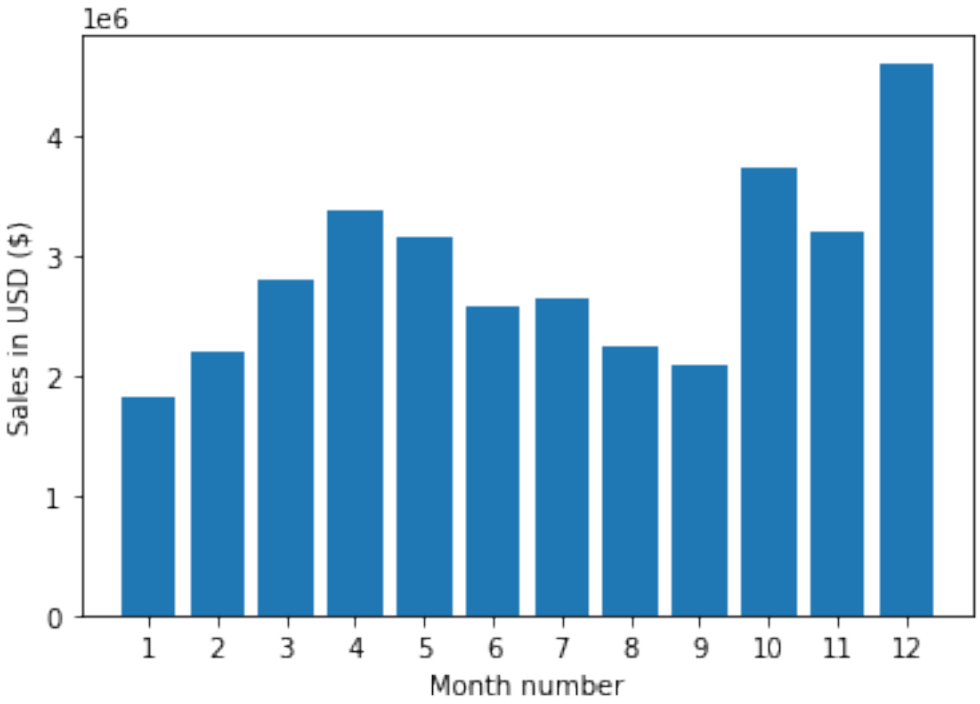
	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City	Sales
0	176558	USB-C Charging Cable	2	11.95	04/19/19 08:46	917 1st St, Dallas, TX 75001	4	Dallas (TX)	23.90
2	176559	Bose SoundSport Headphones	1	99.99	04/07/19 22:30	682 Chestnut St, Boston, MA 02215	4	Boston (MA)	99.99
3	176560	Google Phone	1	600.00	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles (CA)	600.00
4	176560	Wired Headphones	1	11.99	04/12/19 14:38	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles (CA)	11.99
5	176561	Wired Headphones	1	11.99	04/30/19 09:27	333 8th St, Los Angeles, CA 90001	4	Los Angeles (CA)	11.99

```
                City   Sales
0        Dallas (TX)   23.90
2        Boston (MA)   99.99
3   Los Angeles (CA)  600.00
4   Los Angeles (CA)   11.99
5   Los Angeles (CA)   11.99
```

Question 1 :- What was the best month for sales ? How much was earned that month ?

[10]:
```
results = all_data.groupby('Month').sum()
```

[11]:
```
import matplotlib.pyplot as plt
months = range(1,13)
plt.bar(months, results['Sales'])
plt.xticks(months)
plt.ylabel('Sales in USD ($)')
plt.xlabel('Month number')
plt.show()
```



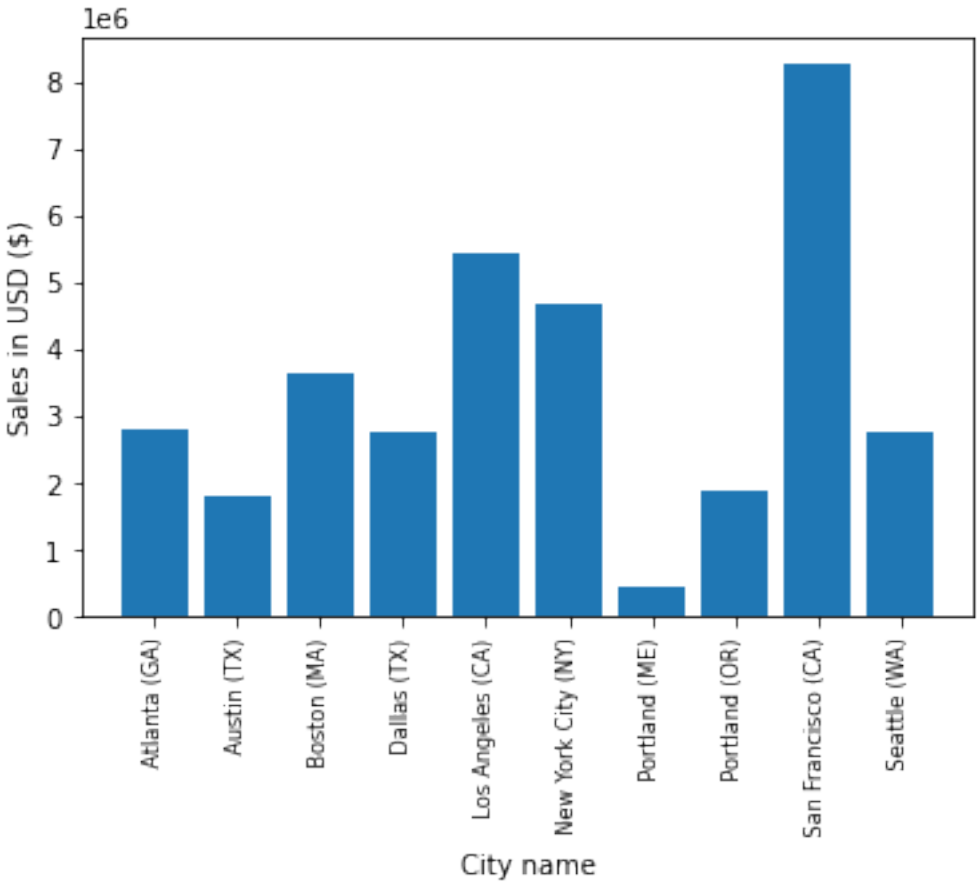
Question 2 :- Which city had the highest number of sales ?

[12]:
```
results = all_data.groupby('City').sum()
results
```

[12]:

City	Quantity Ordered	Price Each	Month	Sales
Atlanta (GA)	16602	2779908.20	104794	2795498.58
Austin (TX)	11153	1809873.61	69829	1819581.75
Boston (MA)	22528	3637409.77	141112	3661642.01
Dallas (TX)	16730	2752627.82	104620	2767975.40
Los Angeles (CA)	33289	5421435.23	208325	5452570.80
New York City (NY)	27932	4635370.83	175741	4664317.43
Portland (ME)	2750	447189.25	17144	449758.27
Portland (OR)	11303	1860558.22	70621	1870732.34
San Francisco (CA)	50239	8211461.74	315520	8262203.91
Seattle (WA)	16553	2733296.01	104941	2747755.48

[13]:
```
import matplotlib.pyplot as plt
Cities = [city for city, df in all_data.groupby('City')]

plt.bar(Cities, results['Sales'])
plt.xticks(Cities, rotation= 'vertical', size=8)
plt.ylabel('Sales in USD ($)')
plt.xlabel('City name')
plt.show()
```



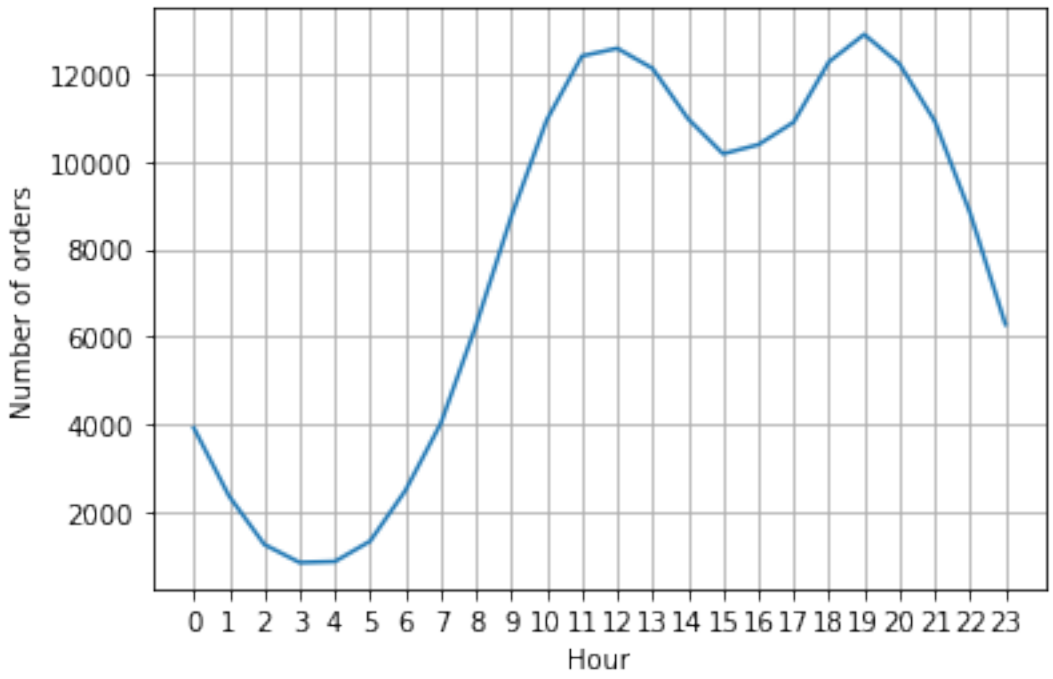
Question 3 :- What time should we display advertisements to maximize likelihood of customer's buying product ?

[14]:
```
all_data['Order Date'] = pd.to_datetime(all_data['Order Date'])
all_data['Hour'] = all_data['Order Date'].dt.hour
all_data['Minute'] = all_data['Order Date'].dt.minute
all_data.head()
```

[14]:

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	City	Sales	Hour	Minute
0	176558	USB-C Charging Cable	2	11.95	2019-04-19 08:46:00	917 1st St, Dallas, TX 75001	4	Dallas (TX)	23.90	8	46
2	176559	Bose SoundSport Headphones	1	99.99	2019-04-07 22:30:00	682 Chestnut St, Boston, MA 02215	4	Boston (MA)	99.99	22	30
3	176560	Google Phone	1	600.00	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles (CA)	600.00	14	38
4	176560	Wired Headphones	1	11.99	2019-04-12 14:38:00	669 Spruce St, Los Angeles, CA 90001	4	Los Angeles (CA)	11.99	14	38
5	176561	Wired Headphones	1	11.99	2019-04-30 09:27:00	333 8th St, Los Angeles, CA 90001	4	Los Angeles (CA)	11.99	9	27

[40]:
```
import matplotlib.pyplot as plt

hours = [hour for hour, df in all_data.groupby('Hour')]
order_counts = all_data.groupby(['Hour']).size()  # Use size() to count the␣
 ↪number of orders

plt.plot(hours, order_counts)
plt.xticks(hours)
plt.xlabel('Hour')
plt.ylabel('Number of orders')
plt.grid()
plt.show()
```



Question 4 :- Which product are most often to sold together ?

[36]:
```
df = all_data[all_data['Order ID'].duplicated(keep=False)]
df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x: ','.
 ↪join(x))
df = df[['Order ID', 'Grouped']].drop_duplicates()
df.head(100)
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_22576\1178528920.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Grouped'] = df.groupby('Order ID')['Product'].transform(lambda x:
','.join(x))
```

[36]:

	Order ID	Grouped
3	176560	Google Phone,Wired Headphones
18	176574	Google Phone,USB-C Charging Cable
30	176585	Bose SoundSport Headphones,Bose SoundSport Hea…
32	176586	AAA Batteries (4-pack),Google Phone
119	176672	Lightning Charging Cable,USB-C Charging Cable
…	…	…

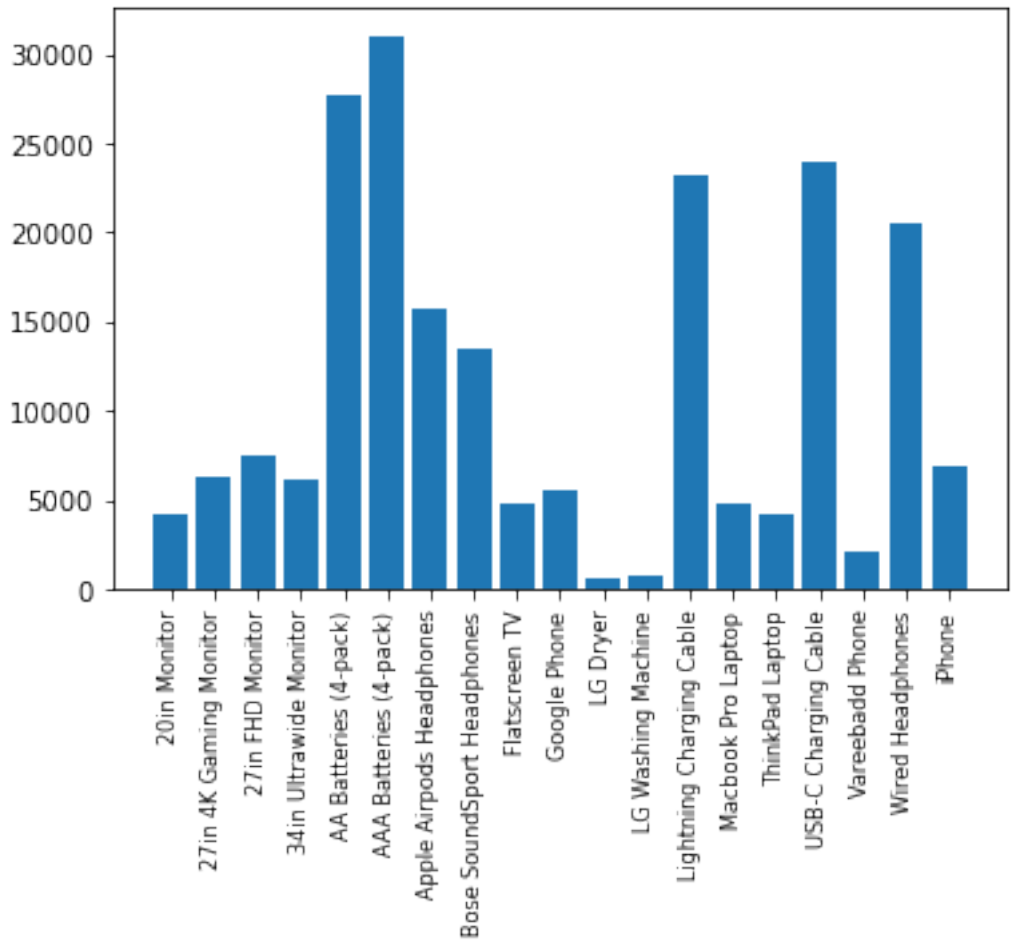
```
2662   179108    Lightning Charging Cable,AAA Batteries (4-pack)
2683   179128                     iPhone,Apple Airpods Headphones
2718   179162                  Google Phone,USB-C Charging Cable
2783   179226          34in Ultrawide Monitor,Macbook Pro Laptop
2829   179270                     iPhone,Lightning Charging Cable

[100 rows x 2 columns]
```

Question 5:- Which product sold the most ? Why do you think it sold the most ?

[39]:
```
from itertools import combinations
from collections import Counter

count = Counter()

# Assuming df is your DataFrame with a 'Grouped' column
for row in df['Grouped']:
    row_list = row.split(',')
    count.update(Counter(combinations(row_list, 2)))

for key, value in count.most_common(10):
    print(key, value)
```

```
('iPhone', 'Lightning Charging Cable') 1005
('Google Phone', 'USB-C Charging Cable') 987
('iPhone', 'Wired Headphones') 447
('Google Phone', 'Wired Headphones') 414
('Vareebadd Phone', 'USB-C Charging Cable') 361
('iPhone', 'Apple Airpods Headphones') 360
('Google Phone', 'Bose SoundSport Headphones') 220
('USB-C Charging Cable', 'Wired Headphones') 160
('Vareebadd Phone', 'Wired Headphones') 143
('Lightning Charging Cable', 'Wired Headphones') 92
```

[27]:
```
product_group = all_data.groupby('Product')
quantity_ordered = product_group.sum()['Quantity Ordered']

keys = [pair for pair, df in product_group]
plt.bar(keys, quantity_ordered)
plt.xticks(keys, rotation='vertical', size=8)
plt.show()
```



[ ]:

[29]:
```
import matplotlib.pyplot as plt

# Assuming you have already defined 'keys', 'quantity_ordered', and 'all_data'

# Create a figure and two subplots (ax1 and ax2)
fig, ax1 = plt.subplots()

# Create a twin axes that shares the same x-axis as ax1
ax2 = ax1.twinx()

# Now you can plot on ax1 and ax2
ax1.bar(keys, quantity_ordered, color='g')
ax2.plot(keys, prices, color='b')

ax1.set_xlabel('Product Name')
```

```
ax1.set_ylabel('Quantity Ordered', color='g')
ax2.set_ylabel('Price ($)', color='b')
ax1.set_xticklabels(keys, rotation='vertical', size=8)

plt.show()  # Display the plot
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_22576\4137774311.py:18: UserWarning:
FixedFormatter should only be used together with FixedLocator
  ax1.set_xticklabels(keys, rotation='vertical', size=8)
```