COMPREHENSIVE SALES ANALYSIS USING PYTHON

SANDHYA HOSKOTE VISWANATHA
M.S. Business Analytics

Project Description

This project aims to process and analyze 12 months of sales data from various sources. It demonstrates the power of pandas for data manipulation, analysis, and visualization in a real-world business context.

Key aspects of the project include:

1. Data Consolidation: Merging multiple CSV files into a single comprehensive dataset.
2. Data Cleaning: Handling missing values and converting data types for analysis.
3. Feature Engineering: Creating new columns to extract additional insights from the data.
4. Data Analysis: Performing various analyses on sales trends, product performance, and customer behavior.
5. Data Visualization: Using matplotlib to create visual representations of the findings.

Data Source

The dataset used in this project is sourced from Keith Galli's GitHub repository, providing a rich set of sales data across different products, locations, and time periods.

Libraries and Dependencies

The project utilizes the following Python libraries:

- pandas: For data manipulation and analysis
- os: For file and directory operations
- matplotlib: For creating data visualizations
- itertools.combinations: Generates all possible combinations of a specified length from a list, tuple, etc.
- collections.Counter: Counts the occurrences of elements in a list, tuple, etc.

Implementation

This project showcases practical applications of data analysis techniques using popular Python libraries, offering insights into sales patterns and trends that can inform business decisions.

1) Importing required libraries.

⇒ The pandas library is used to extract and transform data like reading .csv files, creating data frames, removing nulls, type conversion, grouping and aggregating data, writing into a file, etc.

⇒ The os library allows us to interact with the operating system to access files, create directories, etc.

⇒ The matplotlib is a visualization library used to create charts and graphs with options to customize axes, colors, text, etc., on the visualizations.

```
import pandas as pd
import os
import matplotlib.pyplot as plt
```

2) Extracting and merging data from 12 different .csv files, each containing sales of a particular month in the year 2019. Creating a new single file to be read for further analysis.

Merging 12 months of sales data into a single csv file.

```
# create an empty dataframe to merge all files
all_months_sales=pd.DataFrame()

#create a list of file names in the directory
files=[file for file in os.listdir('./Sales_Data')]

#read each file and place it in the empty dataframe: all_months_sales
for file in files:
    df=pd.read_csv('./Sales_Data/'+file)
    all_months_sales = pd.concat([all_months_sales,df])

#verify if merged
all_months_sales.to_csv('all_sales.csv',index=False)
```

3) Reading the merged file and inspecting the format of data.

Read updated merged data file

```
all_data=pd.read_csv('all_sales.csv')
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	295665	Macbook Pro Laptop	1	1700	12/30/19 00:01	136 Church St, New York City, NY 10001
1	295666	LG Washing Machine	1	600.0	12/29/19 07:03	562 2nd St, New York City, NY 10001
2	295667	USB-C Charging Cable	1	11.95	12/12/19 18:21	277 Main St, New York City, NY 10001
3	295668	27in FHD Monitor	1	149.99	12/22/19 15:13	410 6th St, San Francisco, CA 94016
4	295669	USB-C Charging Cable	1	11.95	12/18/19 12:38	43 Hill St, Atlanta, GA 30301

```
all_data.shape
```

```
(186850, 6)
```

4) Pre-processing the data.

Initial data processing

Data Cleansing

```
#checking for atleast one NaN in all rows
nan_df=all_data[all_data.isna().any(axis=1)]
nan_df.head()

#dropping rows with all NaN
all_data=all_data.dropna(how='all')
all_data.shape
```

```
(186305, 6)
```

```
#based on error faced while converting months to int, there was a value with 'Or'. Some rows had the column headers as values.
#finding such rows.
temp_df=all_data[all_data['Order Date'].str[0:2]=='Or']
temp_df
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
254	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
705	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
1101	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
2875	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
3708	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
...	...	...	...	...	...	...
183671	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
184012	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
184041	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
184275	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
186532	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address

```
#choosing only rows without headers as values
all_data=all_data[all_data['Order Date'].str[0:2]!='Or']
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	295665	Macbook Pro Laptop	1	1700	12/30/19 00:01	136 Church St, New York City, NY 10001
1	295666	LG Washing Machine	1	600.0	12/29/19 07:03	562 2nd St, New York City, NY 10001
2	295667	USB-C Charging Cable	1	11.95	12/12/19 18:21	277 Main St, New York City, NY 10001
3	295668	27in FHD Monitor	1	149.99	12/22/19 15:13	410 6th St, San Francisco, CA 94016
4	295669	USB-C Charging Cable	1	11.95	12/18/19 12:38	43 Hill St, Atlanta, GA 30301

```
#checking data types and converting to appropriate datatypes
all_data.dtypes
```

```
Order ID            object
Product             object
Quantity Ordered    object
Price Each          object
Order Date          object
Purchase Address    object
dtype: object
```

```
all_data['Quantity Ordered']=pd.to_numeric(all_data['Quantity Ordered'])
all_data['Price Each']=pd.to_numeric(all_data['Price Each'])

all_data.dtypes
```

```
Order ID             object
Product              object
Quantity Ordered      int64
Price Each          float64
Order Date           object
Purchase Address     object
dtype: object
```

```
all_data['Order Date']=pd.to_datetime(all_data['Order Date'])
all_data.dtypes
```

```
/var/folders/pf/2vkl493s1q593h7prmj8s4j80000gp/T/ipykernel_42195/2924847002.py:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  all_data['Order Date']=pd.to_datetime(all_data['Order Date'])
```

```
Order ID                    object
Product                     object
Quantity Ordered             int64
Price Each                 float64
Order Date          datetime64[ns]
Purchase Address            object
dtype: object
```

```
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016
4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301

Creating columns in the data that would aid the analysis either by splitting existing columns, extracting date parts, or creating new calculations.

Data Formulation

```
#augment data with additional columns

#adding a month column
all_data['Month']=all_data['Order Date'].dt.month
all_data['Month']=all_data['Month'].astype('int')
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12
4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	12

```
#adding a sales column
all_data['Sales']=all_data['Quantity Ordered']*all_data['Price Each']
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99
4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	12	11.95

```
#adding a city column
all_data['City'] = all_data['Purchase Address'].str.split(',').str[1].str.strip()
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00	New York City
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00	New York City
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95	New York City
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco
4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	12	11.95	Atlanta

```
#adding a state column using apply and lambda function
def get_state(address):
    return address.split(',')[2].split(' ')[1]
all_data['State']=all_data['Purchase Address'].apply(lambda x:get_state(x))

all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	State
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00	New York City	NY
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00	New York City	NY
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95	New York City	NY
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco	CA

```
#Some cities may have same names but in different states. Creating a column to concatenate city+State for clarity
all_data['City (State)']=all_data['City']+' ('+all_data['State']+')'
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	State	City (State)
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00	New York City	NY	New York City (NY)
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00	New York City	NY	New York City (NY)
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95	New York City	NY	New York City (NY)
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco	CA	San Francisco (CA)
4	295669	USB-C Charging Cable	1	11.95	2019-12-18 12:38:00	43 Hill St, Atlanta, GA 30301	12	11.95	Atlanta	GA	Atlanta (GA)

```
#Extracting hours from datetime
all_data['Hour']=all_data['Order Date'].dt.hour
all_data.head()
```

	Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address	Month	Sales	City	State	City (State)	Hour
0	295665	Macbook Pro Laptop	1	1700.00	2019-12-30 00:01:00	136 Church St, New York City, NY 10001	12	1700.00	New York City	NY	New York City (NY)	0
1	295666	LG Washing Machine	1	600.00	2019-12-29 07:03:00	562 2nd St, New York City, NY 10001	12	600.00	New York City	NY	New York City (NY)	7
2	295667	USB-C Charging Cable	1	11.95	2019-12-12 18:21:00	277 Main St, New York City, NY 10001	12	11.95	New York City	NY	New York City (NY)	18
3	295668	27in FHD Monitor	1	149.99	2019-12-22 15:13:00	410 6th St, San Francisco, CA 94016	12	149.99	San Francisco	CA	San Francisco (CA)	15

5) Analyzing the data to derive insights.

Analysis

Insight 1: Show monthly sales

```
result1=all_data.groupby('Month')['Sales'].sum()
result1
```

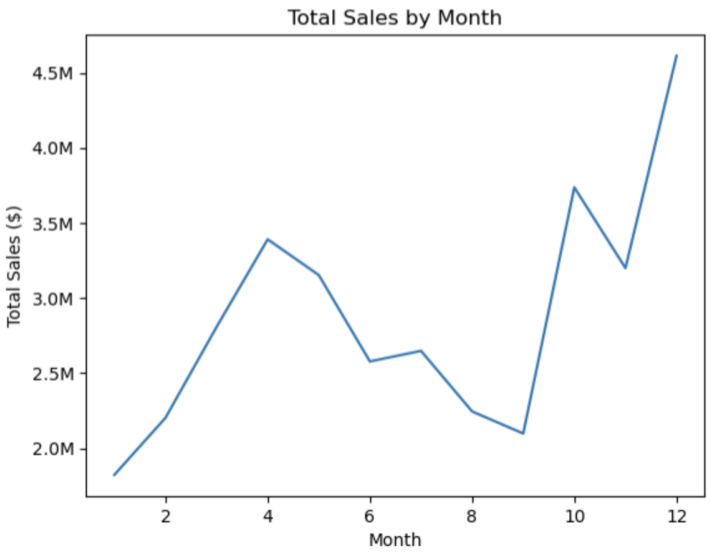
```
Month
1     1822256.73
2     2202022.42
3     2807100.38
4     3390670.24
5     3152606.75
6     2577802.26
7     2647775.76
8     2244467.88
9     2097560.13
10    3736726.88
11    3199603.20
12    4613443.34
Name: Sales, dtype: float64
```

```
from matplotlib.ticker import FuncFormatter

# a function to format axis ticks in millions
def millions(x, pos):
    """Format tick values to display in millions."""
    return f'{x / 1e6:.1f}M'
```

```
fig, ax = plt.subplots()
result1.plot(kind='line',ax=ax)

# Apply the custom formatter to the y-axis
ax.yaxis.set_major_formatter(FuncFormatter(millions))
plt.title('Total Sales by Month')
plt.xlabel('Month')
plt.ylabel('Total Sales ($)')
plt.show()
```



> Sales peak during the holiday season (October-December) and decline in the early months, likely due to holiday spending and post-holiday budget constraints. A line chart helps recognize this trend in the best manner.

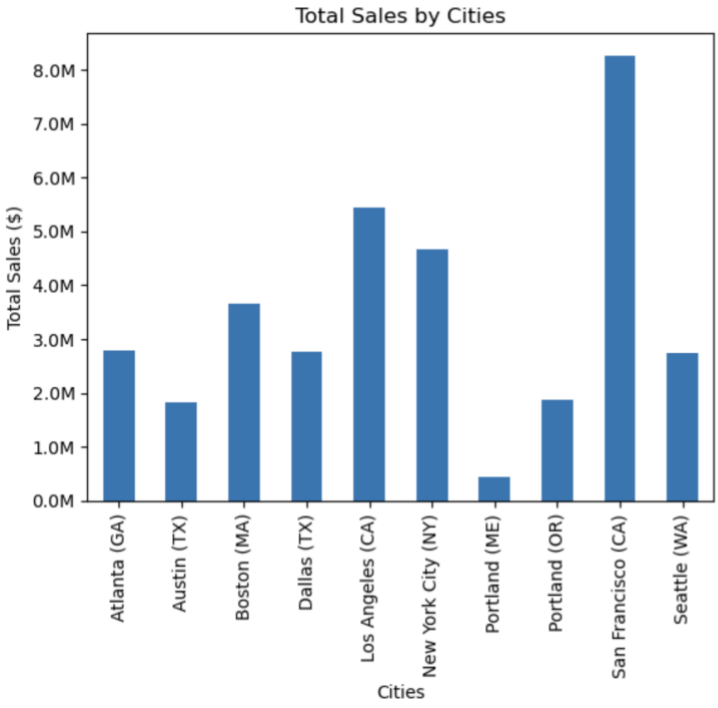
Insight 2: Which city had the most sales?

```
result2=all_data.groupby('City (State)')['Sales'].sum()
result2
```

```
City (State)
Atlanta (GA)          2795498.58
Austin (TX)           1819581.75
Boston (MA)           3661642.01
Dallas (TX)           2767975.40
Los Angeles (CA)      5452570.80
New York City (NY)    4664317.43
Portland (ME)          449758.27
Portland (OR)         1870732.34
San Francisco (CA)    8262203.91
Seattle (WA)          2747755.48
Name: Sales, dtype: float64
```

```
fig, ax=plt.subplots()
result2.plot(kind='bar',ax=ax)

ax.yaxis.set_major_formatter(FuncFormatter(millions))
plt.title('Total Sales by Cities')
plt.xlabel('Cities')
plt.ylabel('Total Sales')
plt.show()
```



> California ranks #1 in total sales, while Portland-Maine ranks last. A bar chart helps visualize this categorical data, with height being the visual cue for the magnitude of sales.

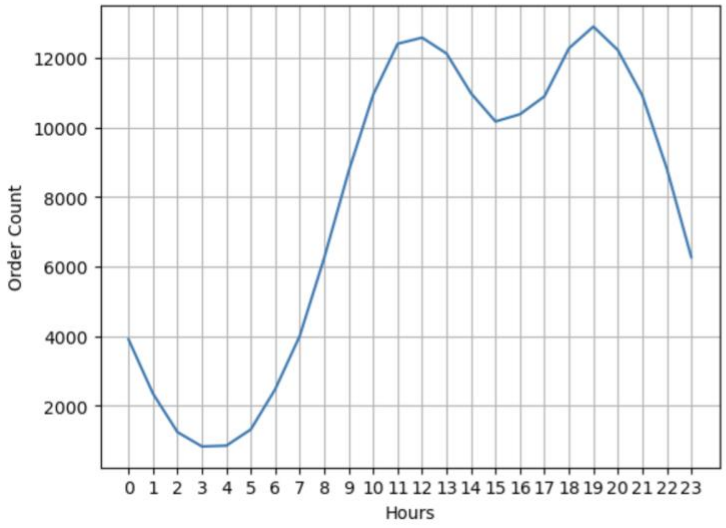
Insight 3: What time should we display advertisements to maximize likelihood of customers buying products?

```
result3=all_data.groupby('Hour')['Order ID'].count()
result3
```

```
Hour
0      3910
1      2350
2      1243
3       831
4       854
5      1321
6      2482
7      4011
8      6256
9      8748
10    10944
11    12411
12    12587
13    12129
14    10984
15    10175
16    10384
17    10899
18    12280
19    12905
20    12228
21    10921
22     8822
23     6275
Name: Order ID, dtype: int64
```

```
fig, ax= plt.subplots()
hours=range(0,24)

result3.plot(kind='line',ax=ax)
plt.xticks(hours)
plt.xlabel('Hours')
plt.ylabel('Order Count')
plt.grid()
plt.show()
```



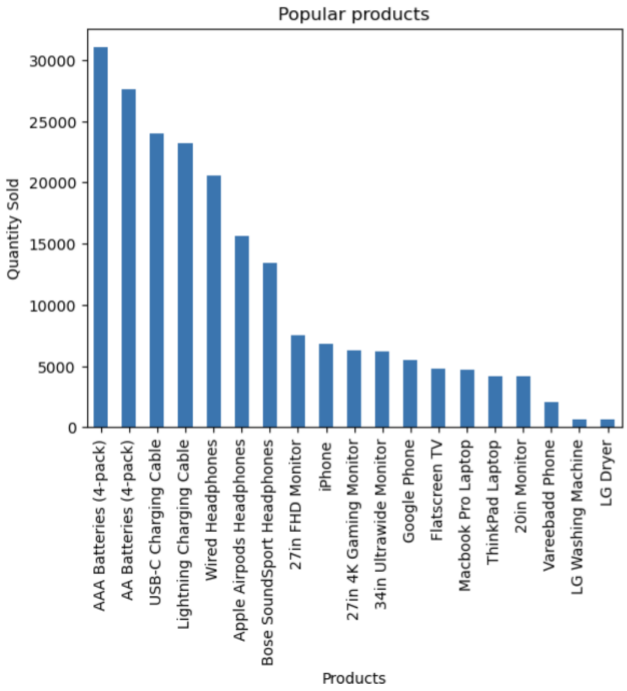
> Consumers tend to place most orders between 11-12 pm and peak at around 7 pm. Just before these times would be the best time during the day to display advertisements to encourage more orders. The line chart with gridlines aids in tracking fluctuations accurately and enhances hourly insight.

Insight 4: What products sold the most?

```
result4=all_data.groupby('Product')['Quantity Ordered'].sum()
result4=result4.sort_values(ascending=False)
result4
```

```
Product
AAA Batteries (4-pack)        31017
AA Batteries (4-pack)         27635
USB-C Charging Cable          23975
Lightning Charging Cable      23217
Wired Headphones              20557
Apple Airpods Headphones      15661
Bose SoundSport Headphones    13457
27in FHD Monitor               7550
iPhone                         6849
27in 4K Gaming Monitor         6244
34in Ultrawide Monitor         6199
Google Phone                   5532
Flatscreen TV                  4819
Macbook Pro Laptop             4728
ThinkPad Laptop                4130
20in Monitor                   4129
Vareebadd Phone                2068
LG Washing Machine              666
LG Dryer                        646
Name: Quantity Ordered, dtype: int64
```

```
fig, ax1 = plt.subplots()
result4.plot(kind='bar',ax=ax1)
plt.title('Popular products')
plt.xlabel('Products')
plt.ylabel('Quantity Sold')
plt.show()
```



> The highest quantity sold product is AAA batteries, and the lowest quantity sold is LG dryers. Sorting the bars helps us identify the ranking of the products better. This can help shift focus to low-selling products and devise better marketing campaigns.

Insight 5: What products are most often sold together?

```
#create a dataframe to group products by order id
df=all_data[all_data['Order ID'].duplicated(keep=False)]

#create a column to create a column separated list of products in the same order
df['Grouped']=df.groupby('Order ID')['Product'].transform(lambda x:','.join(x))

#remove the duplicate columns with the same Order ID and Grouped values
df=df[['Order ID','Grouped']].drop_duplicates()

df.head()
```

```
/var/folders/pf/2vkl493s1q593h7prmj8s4j80000gp/T/ipykernel_42195/698168304.py:5: 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))
```

	Order ID	Grouped
16	295681	Google Phone,USB-C Charging Cable,Bose SoundSp...
36	295698	Vareebadd Phone,USB-C Charging Cable
42	295703	AA Batteries (4-pack),Bose SoundSport Headphones
66	295726	iPhone,Lightning Charging Cable
76	295735	iPhone,Apple Airpods Headphones,Wired Headphones

> After grouping all products that occur in the same order, we have to count such occurrences to find the products that are most often bought together.

```
from itertools import combinations
from collections import Counter
```

```
count=Counter()

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
```

> We present the results to highlight pairs of items (limited to two per order) that are frequently bought together. The analysis reveals that iPhones and lightning charging cables are often purchased together.