Problem Statement:

A grocery store shared the transactional data with you. Your job is to conduct a thorough analysis of Point of Sale (POS) data, identify the most commonly occurring sets of items in the customer orders, and provide recommendations through which a grocery store can increase its revenue by popular combo offers & discounts for customers.

In [ ]:
```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pylab import rcParams
import plotly.express as px

# Setup and imports
import sys
import os
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in /usr/local/lib/python3.12/dist-packages (5.
24.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.12/dist-pac
kages (from plotly) (8.5.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages
(from plotly) (25.0)
```

Exploratory Analysis

In [ ]:
```
#Read the data
gro_df = pd.read_csv('/content/dataset_group.csv')

print("All modules imported successfully!")
print(f"Current working directory: {os.getcwd()}")
```

```
All modules imported successfully!
Current working directory: /content
```

In [ ]:
```
#view the first 10 records
gro_df.head(10)
```

Out[ ]:

	Date	Order_id	Product
0	01-01-2018	1	yogurt
1	01-01-2018	1	pork
2	01-01-2018	1	sandwich bags
3	01-01-2018	1	lunch meat
4	01-01-2018	1	all- purpose
5	01-01-2018	1	flour
6	01-01-2018	1	soda
7	01-01-2018	1	butter
8	01-01-2018	1	beef
9	01-01-2018	1	aluminum foil

In [ ]:

```
#view the last 10 records
gro_df.tail(10)
```

Out[ ]:

	Date	Order_id	Product
20631	25-02-2020	1138	all- purpose
20632	25-02-2020	1138	sandwich bags
20633	25-02-2020	1138	toilet paper
20634	25-02-2020	1138	soda
20635	25-02-2020	1138	soda
20636	25-02-2020	1138	soda
20637	25-02-2020	1138	paper towels
20638	26-02-2020	1139	soda
20639	26-02-2020	1139	laundry detergent
20640	26-02-2020	1139	shampoo

In [ ]:

```
#Display the data types
gro_df.dtypes
```

Out[ ]:

	0
Date	object
Order_id	int64
Product	object

dtype: object

Date and Product are of object data types. Order_id is an integer

In [ ]:
```
#convert Date to a date data type
gro_df['Date'] = pd.to_datetime(gro_df['Date'], format='%d-%m-%Y')

gro_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20641 entries, 0 to 20640
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      20641 non-null  datetime64[ns]
 1   Order_id  20641 non-null  int64
 2   Product   20641 non-null  object
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 483.9+ KB
```

In [ ]:
```
#view the number of records
gro_df.shape
```

Out[ ]:
```
(20641, 3)
```

There are 20641 records and 3 rows

In [ ]:
```
#Check if there are any null values/records
gro_df.isnull().sum()
```

Out[ ]:

	0
Date	0
Order_id	0
Product	0

dtype: int64

There are no null records in the dataset

In [ ]:
```
#check if there are duplicates
gro_df.duplicated().sum()
```

```
Out[ ]: np.int64(4730)
```

There are 4,730 duplicated rows in the DataFrame. This is because different orsers may contain similar items

```
In [ ]: #Generate unique items on the Product column
        gro_df['Product'].unique()
```

```
Out[ ]: array(['yogurt', 'pork', 'sandwich bags', 'lunch meat', 'all- purpose',
               'flour', 'soda', 'butter', 'beef', 'aluminum foil', 'dinner rolls',
               'shampoo', 'mixes', 'soap', 'laundry detergent', 'ice cream',
               'toilet paper', 'hand soap', 'waffles', 'cheeses', 'milk',
               'dishwashing liquid/detergent', 'individual meals', 'cereals',
               'tortillas', 'spaghetti sauce', 'ketchup', 'sandwich loaves',
               'poultry', 'bagels', 'eggs', 'juice', 'pasta', 'paper towels',
               'coffee/tea', 'fruits', 'sugar'], dtype=object)
```

```
In [ ]: #generate a count of unique items in the Product column
        len(gro_df['Product'].unique())
```

```
Out[ ]: 37
```

```
In [ ]: # Count occurrences of each product
        product_counts = gro_df['Product'].value_counts()

        # Display the product counts
        display(product_counts)
```

	count
Product	
poultry	640
soda	597
cereals	591
ice cream	579
cheeses	578
waffles	575
soap	574
lunch meat	573
bagels	573
eggs	570
juice	570
toilet paper	569
dinner rolls	567
aluminum foil	566
coffee/tea	565
shampoo	562
beef	561
paper towels	556
flour	555
butter	555
milk	555
mixes	554
dishwashing liquid/detergent	551
all- purpose	551
ketchup	548
yogurt	545
individual meals	544
tortillas	543
laundry detergent	542

	count
Product	
pasta	542
sandwich bags	536
spaghetti sauce	536
sugar	533
pork	531
fruits	529
sandwich loaves	523
hand soap	502

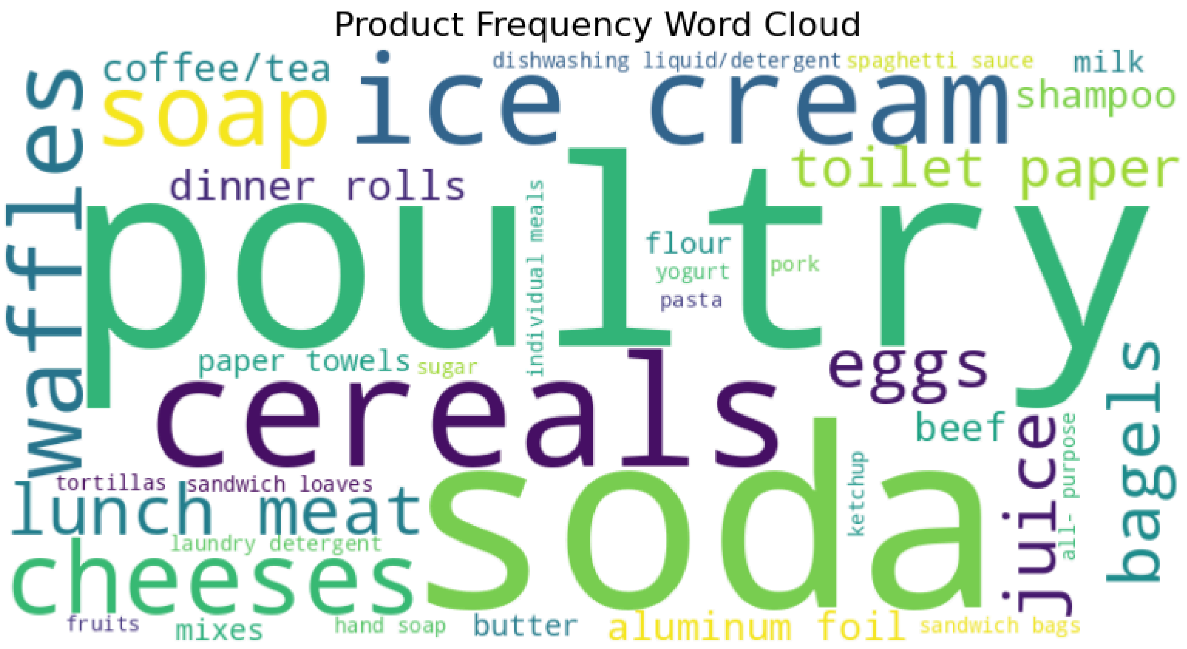
dtype: int64

In [ ]:
```
import pandas as pd
import plotly.express as px

# --- 1. Prepare Data for Plotly ---
# product_counts is a pandas Series (e.g., from gro_df['Product'].value_counts())
# Convert the Series to a DataFrame required by Plotly
product_df = product_counts.reset_index()
product_df.columns = ['Product', 'Count']

# --- 2. Treemap Generation ---
fig = px.treemap(
    product_df,
    # path defines the hierarchy: creates a root "All Products" and then divides by
    path=[px.Constant("All Products"), 'Product'],
    # values determines the size of the rectangles
    values='Count',
    title='Product Frequency Treemap',
    # Color the rectangles based on their count
    color='Count',
    color_continuous_scale='Plasma'
)

# Optional: Adjust margins for better display
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))

# --- 3. Save Output ---
# This saves an interactive HTML file which you can open in your browser
fig.write_html('product_frequency_treemap.html')

print("Treemap generated and saved as 'product_frequency_treemap.html'")
```

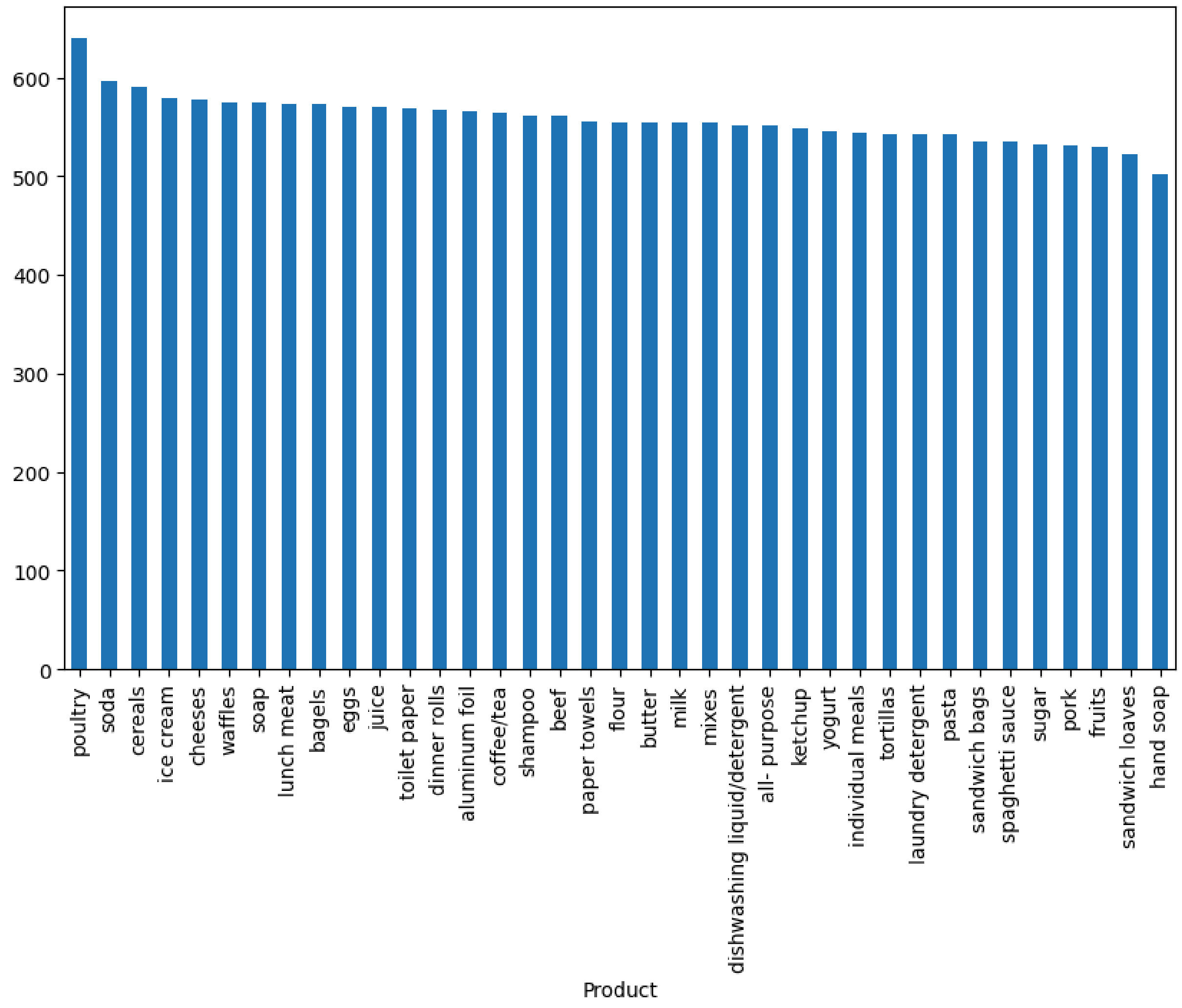
```
Treemap generated and saved as 'product_frequency_treemap.html'
```

In [ ]:
```
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

```
import pandas as pd



# The WordCloud generator requires a dictionary of {word: frequency}
# We convert the pandas Series to a dictionary
wordcloud_data = product_counts.to_dict()

# --- 2. Configure and Generate the Word Cloud ---

# Create a WordCloud object
wc = WordCloud(
    background_color="white",
    max_words=100,  # Limit the number of words displayed
    width=800,      # Set the width of the canvas
    height=400,     # Set the height of the canvas
    colormap="viridis" # Choose a color scheme
)

# Generate the word cloud from the frequency dictionary
wordcloud = wc.generate_from_frequencies(wordcloud_data)

# --- 3. Display the Word Cloud ---

plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")  # Hide the axes
plt.title("Product Frequency Word Cloud", fontsize=16)
plt.show()
```



```
In [ ]: #generate a bar chart of the items above
        product_counts.plot(kind='bar', figsize=(10, 6))

Out[ ]: <Axes: xlabel='Product'>
```



The count of each items is almost equal. Poultry has the highest while hand soap has the least.

Time Series - Yearly

Split the Dates to Year, Quarter, Month and week

In [ ]:
```
#copy the Date filed
df_datesplit = gro_df.copy()
```

In [ ]:
```
df_datesplit['YEAR'] = pd.PeriodIndex(df_datesplit['Date'], freq='Y')
df_datesplit['MONTH'] = pd.PeriodIndex(df_datesplit['Date'], freq='M')
df_datesplit['QUARTER'] = pd.PeriodIndex(df_datesplit['Date'], freq='Q')
df_datesplit['WEEK'] = pd.PeriodIndex(df_datesplit['Date'], freq='W')
```

In [ ]:
```
df_datesplit['DAY'] = df_datesplit.Date.dt.strftime('%A')
df_datesplit['MONTH_NAME'] = df_datesplit['Date'].dt.strftime('%b')
```

In [ ]:
```

```

In [ ]:
```
df_datesplit.head()
```

Out[ ]:

	Date	Order_id	Product	Date_copy	YEAR	MONTH	QUARTER	WEEK	DAY	M
0	2018-01-01	1	yogurt	2018-01-01	2018	2018-01	2018Q1	2018-01-01/2018-01-07	Monday	
1	2018-01-01	1	pork	2018-01-01	2018	2018-01	2018Q1	2018-01-01/2018-01-07	Monday	
2	2018-01-01	1	sandwich bags	2018-01-01	2018	2018-01	2018Q1	2018-01-01/2018-01-07	Monday	
3	2018-01-01	1	lunch meat	2018-01-01	2018	2018-01	2018Q1	2018-01-01/2018-01-07	Monday	
4	2018-01-01	1	all-purpose	2018-01-01	2018	2018-01	2018Q1	2018-01-01/2018-01-07	Monday	

In [ ]:
```
df_datesplit.tail()
```

Out[ ]:

	Date	Order_id	Product	Date_copy	YEAR	MONTH	QUARTER	WEEK	
20636	2020-02-25	1138	soda	2020-02-25	2020	2020-02	2020Q1	2020-02-24/2020-03-01	Tue
20637	2020-02-25	1138	paper towels	2020-02-25	2020	2020-02	2020Q1	2020-02-24/2020-03-01	Tue
20638	2020-02-26	1139	soda	2020-02-26	2020	2020-02	2020Q1	2020-02-24/2020-03-01	Wedne
20639	2020-02-26	1139	laundry detergent	2020-02-26	2020	2020-02	2020Q1	2020-02-24/2020-03-01	Wedne
20640	2020-02-26	1139	shampoo	2020-02-26	2020	2020-02	2020Q1	2020-02-24/2020-03-01	Wedne

In [ ]:

```
#aggregate products per year
df3 = df_datesplit.copy()
df_yr_products = pd.DataFrame(df3[['YEAR', 'Order_id', 'Product']].groupby('YEAR').
                                                                  'Product': 'count
newname = {'Product': 'Num_of_Products'}
df_yr_products.rename(columns=newname, inplace=True)
df_yr_products
```

Out[ ]:

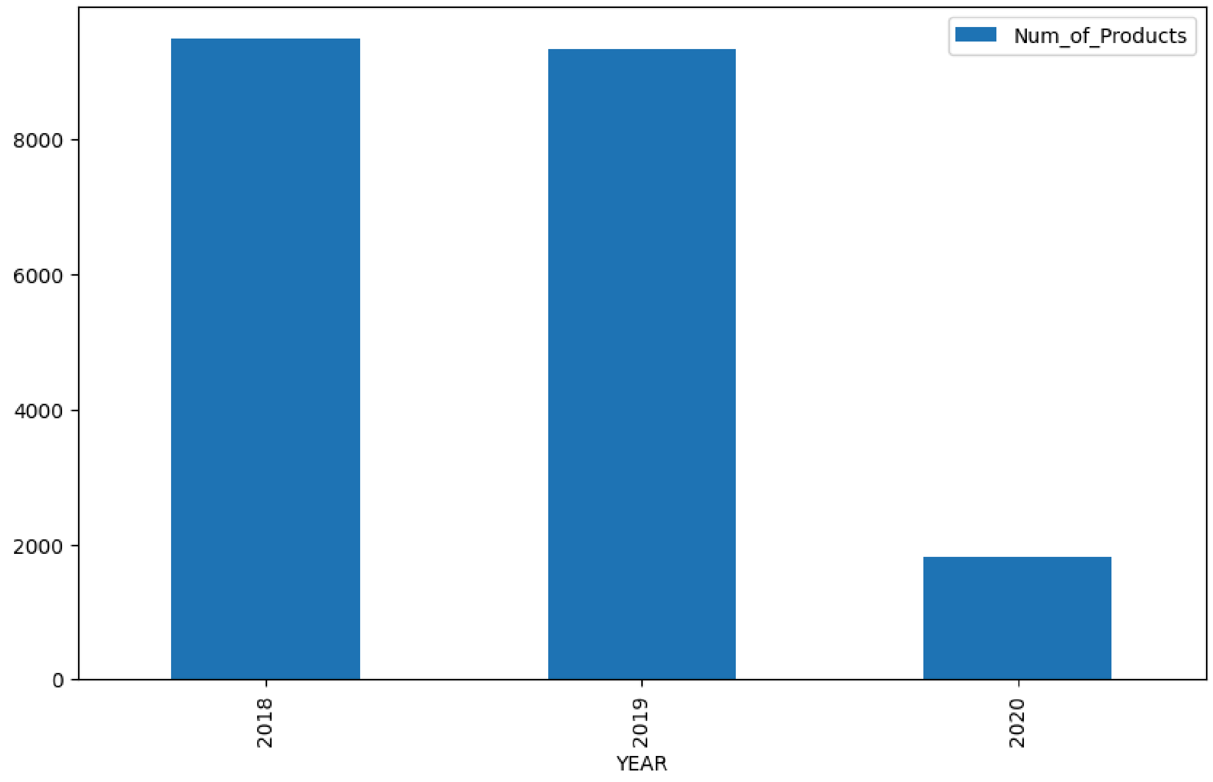
	Num_of_Products
YEAR	
2018	9479
2019	9333
2020	1829

In [ ]:

```
#generate a bar chart for the data above
df_yr.plot(kind='bar', figsize=(10, 6))
```

Out[ ]:

```
<Axes: xlabel='YEAR'>
```

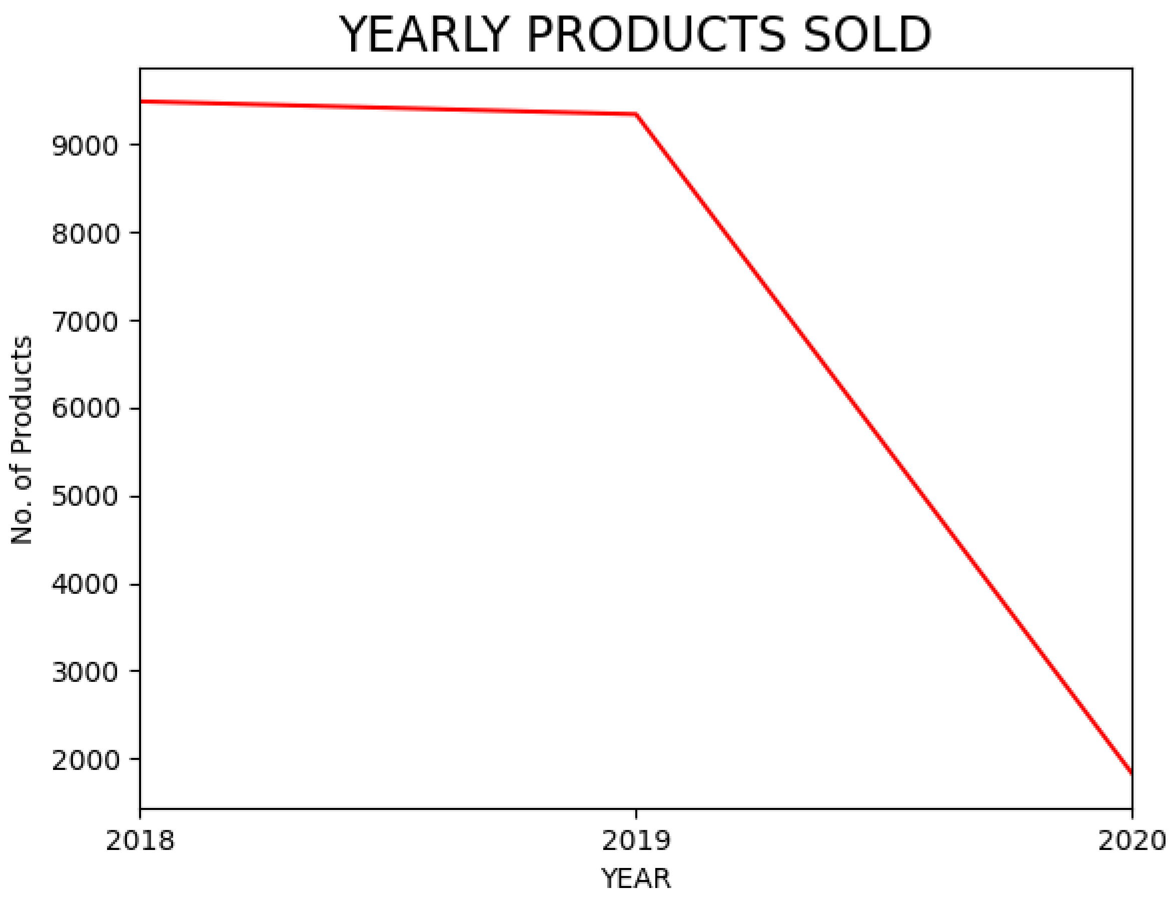


In [ ]:
```
#aggregate orders per year
df3 = df_datesplit.copy()
df_yr = pd.DataFrame(df3[['YEAR', 'Order_id']].groupby('YEAR').agg({'Order_id': 'nu
newname = {'Order_id': 'Num_of_Orders'}
df_yr.rename(columns=newname, inplace=True)
df_yr
```

Out[ ]:

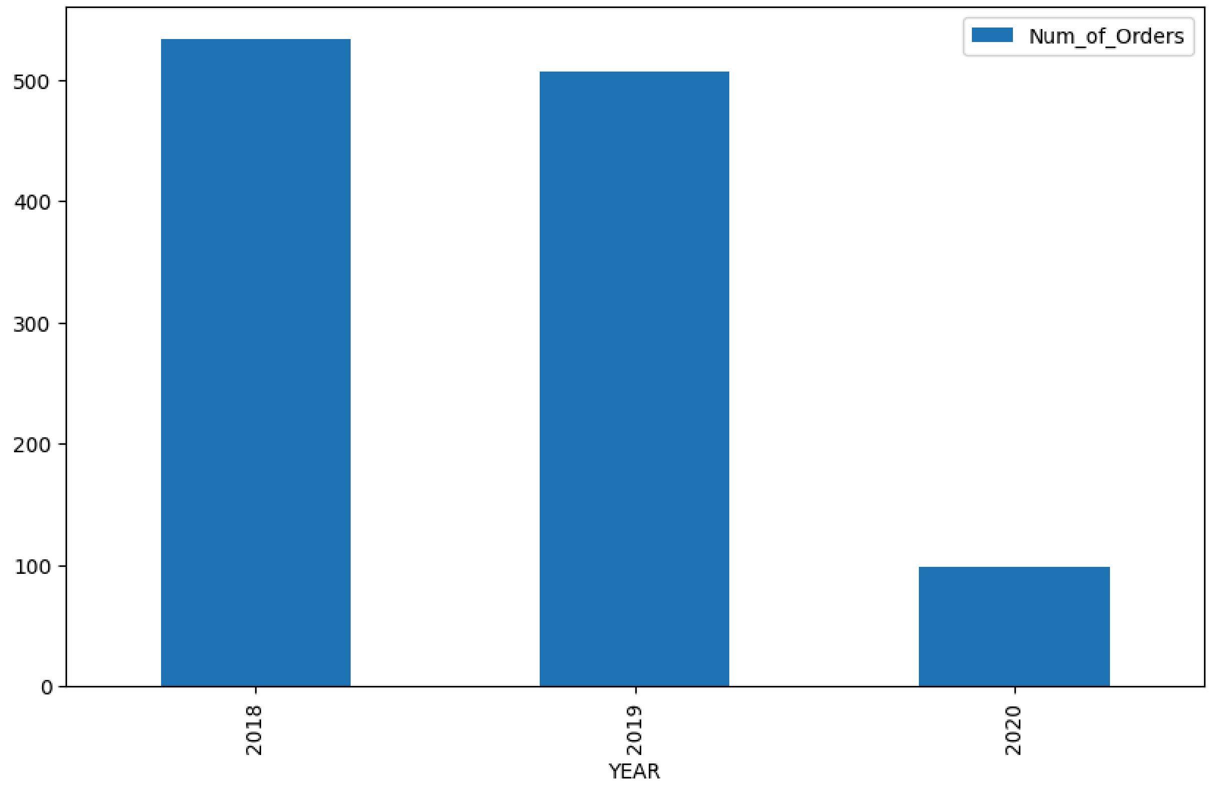
YEAR	Num_of_Orders
2018	533
2019	507
2020	99

In [ ]:
```
df_yr_products['Num_of_Products'].plot(color='red');
plt.ylabel('No. of Products');
plt.title('YEARLY PRODUCTS SOLD', fontsize=17)
plt.savefig('ts_YEAR_prod.jpg', bbox_inches='tight');
```

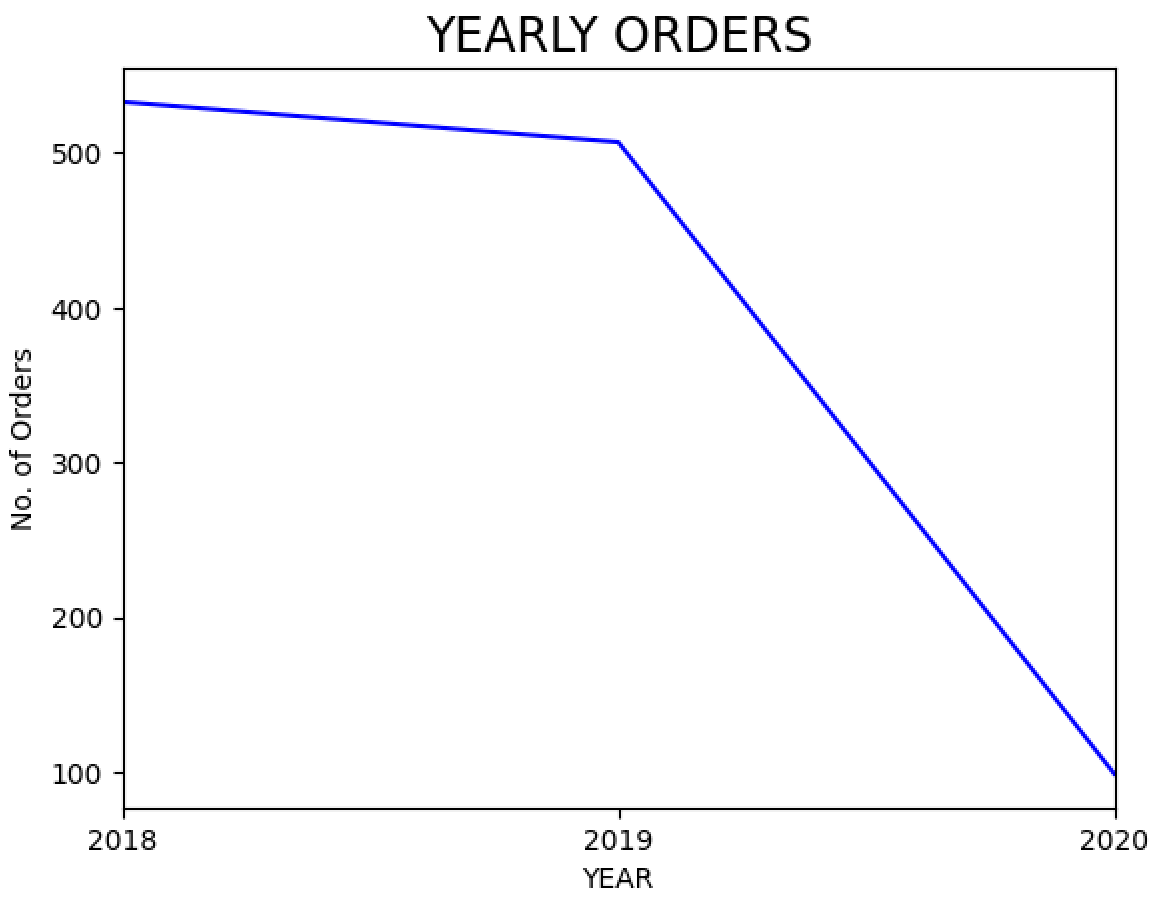


```
In [ ]: #Generate a bar chart for the data above
        df_yr.plot(kind='bar', figsize=(10, 6))
```

```
Out[ ]: <Axes: xlabel='YEAR'>
```



```
In [ ]: df_yr['Num_of_Orders'].plot(color='blue');
        plt.ylabel('No. of Orders');
        plt.title('YEARLY ORDERS', fontsize=17)
        plt.savefig('ts_YEAR_order.jpg', bbox_inches='tight');
```



Time Series - Quarterly

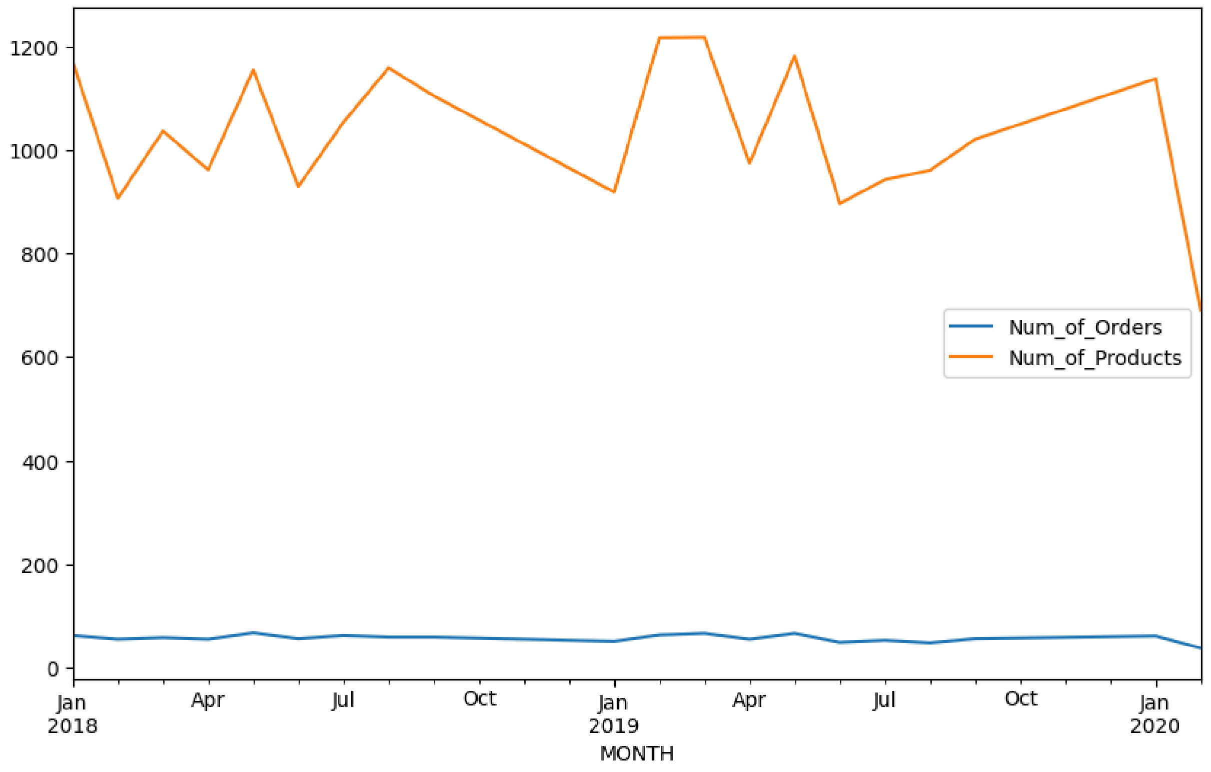
In [ ]:
```
df3 = df_datesplit.copy()
df_month = pd.DataFrame(df3[['MONTH', 'Order_id', 'Product']].groupby('MONTH').agg(
                                                                    'Product': 'count
newname = {'Order_id': 'Num_of_Orders', 'Product': 'Num_of_Products'}
df_month.rename(columns=newname, inplace=True)
df_month
```

Out[ ]:

MONTH	Num_of_Orders	Num_of_Products
2018-01	62	1170
2018-02	55	907
2018-03	58	1037
2018-04	55	962
2018-05	67	1155
2018-06	56	930
2018-07	62	1054
2018-08	59	1159
2018-09	59	1105
2019-01	51	919
2019-02	63	1217
2019-03	66	1218
2019-04	55	975
2019-05	66	1182
2019-06	49	897
2019-07	53	943
2019-08	48	961
2019-09	56	1021
2020-01	61	1138
2020-02	38	691

In [ ]:
```
#generate line chart for the Num_of_Orders and Num_of_Products
df_month.plot(figsize=(10, 6))
```

Out[ ]:
```
<Axes: xlabel='MONTH'>
```



Time Series - Monthly

```
In [ ]: df3 = df_datesplit.copy()
        df_month = pd.DataFrame(df3[['MONTH', 'Order_id', 'Product']].groupby('MONTH').agg(
                                                                     'Product': 'count
        newname = {'Order_id': 'Num_of_Orders', 'Product': 'Num_of_Products'}
        df_month.rename(columns=newname, inplace=True)
        df_month
```

Out[ ]:

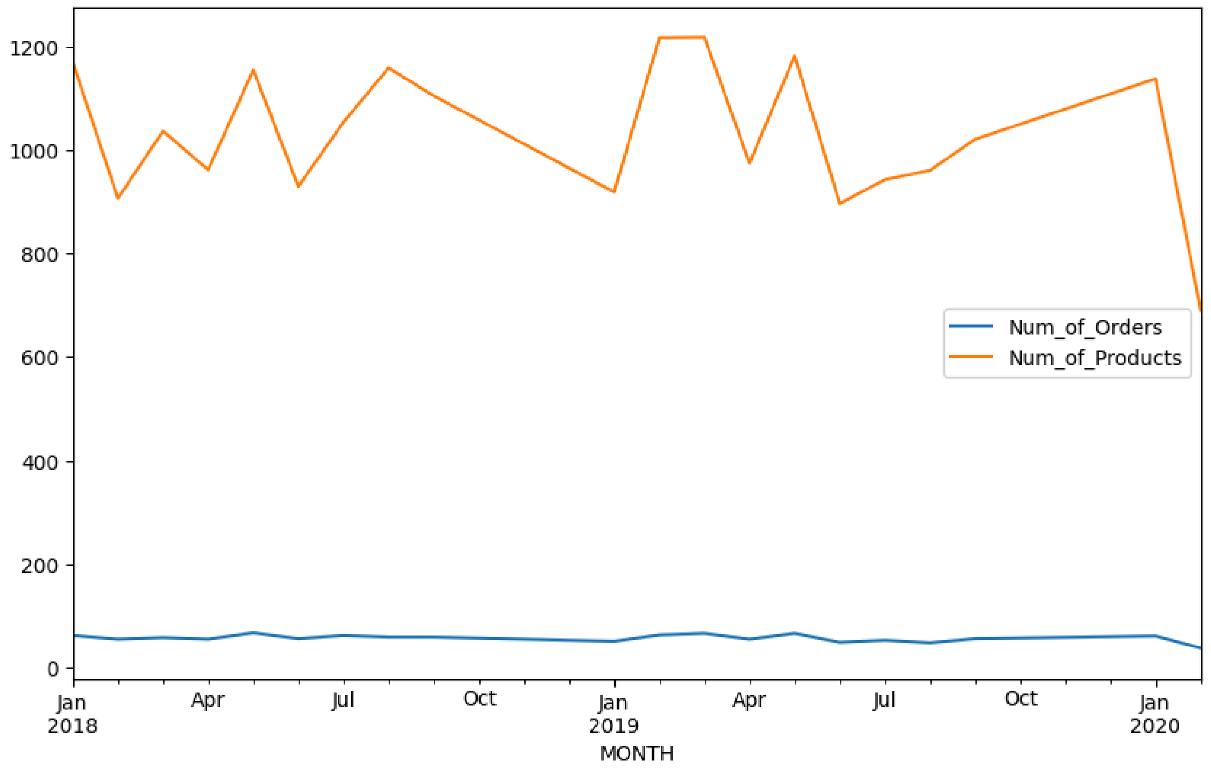
MONTH	Num_of_Orders	Num_of_Products
2018-01	62	1170
2018-02	55	907
2018-03	58	1037
2018-04	55	962
2018-05	67	1155
2018-06	56	930
2018-07	62	1054
2018-08	59	1159
2018-09	59	1105
2019-01	51	919
2019-02	63	1217
2019-03	66	1218
2019-04	55	975
2019-05	66	1182
2019-06	49	897
2019-07	53	943
2019-08	48	961
2019-09	56	1021
2020-01	61	1138
2020-02	38	691

In [ ]:

```
#generate line chart for the Num_of_Orders and Num_of_Products
df_month.plot(figsize=(10, 6))
```

Out[ ]:

```
<Axes: xlabel='MONTH'>
```



Time Series - Weekly

```
In [ ]: df3 = df_datesplit.copy()
        df_week = pd.DataFrame(df3[['WEEK', 'Order_id', 'Product']].groupby('WEEK').agg({'O
                                                                      'Product': 'count
        newname = {'Order_id': 'Num_of_Orders', 'Product': 'Num_of_Products'}
        df_week.rename(columns=newname, inplace=True)
        df_week
```

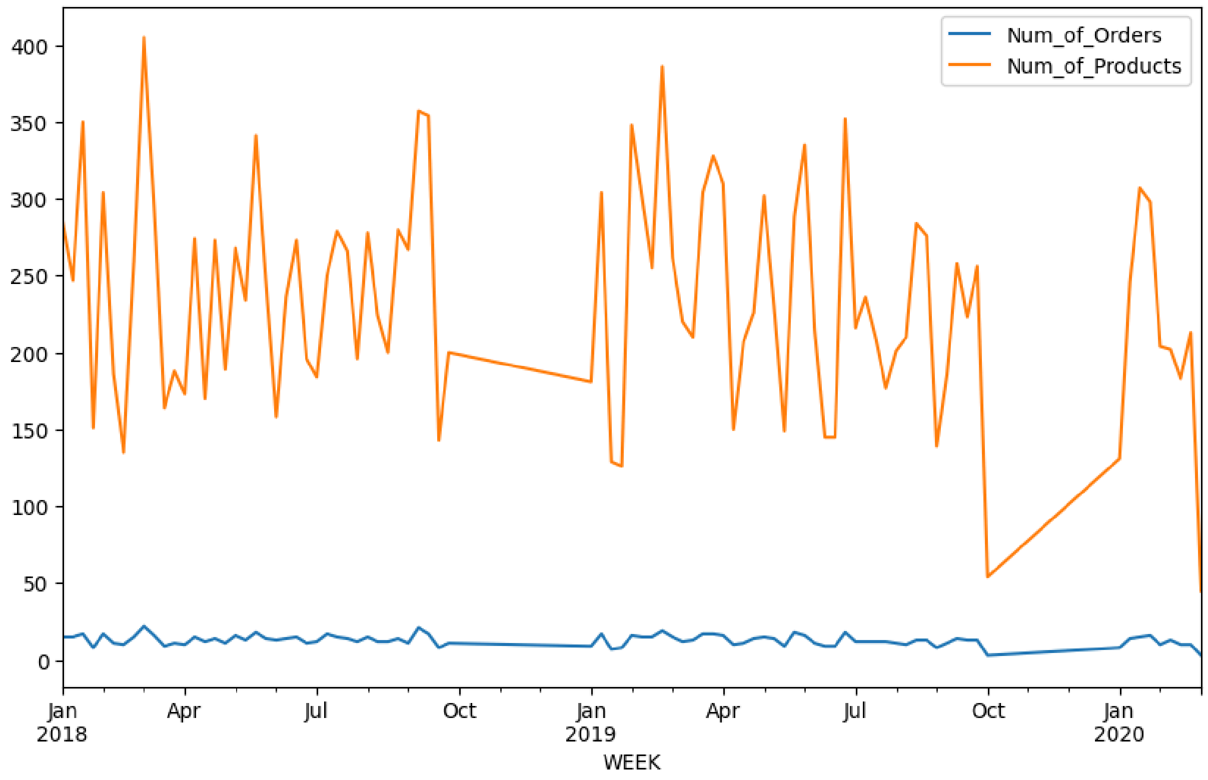
Out[ ]:

WEEK	Num_of_Orders	Num_of_Products
2018-01-01/2018-01-07	15	285
2018-01-08/2018-01-14	15	247
2018-01-15/2018-01-21	17	350
2018-01-22/2018-01-28	8	151
2018-01-29/2018-02-04	17	304
...	...	...
2020-01-27/2020-02-02	10	204
2020-02-03/2020-02-09	13	202
2020-02-10/2020-02-16	10	183
2020-02-17/2020-02-23	10	213
2020-02-24/2020-03-01	3	45

88 rows × 2 columns

In [ ]:
```python
#generate line chart for the Num_of_Orders and Num_of_Products
df_week.plot(figsize=(10, 6))
```

Out[ ]:
```
<Axes: xlabel='WEEK'>
```



Time Series - Days of the week

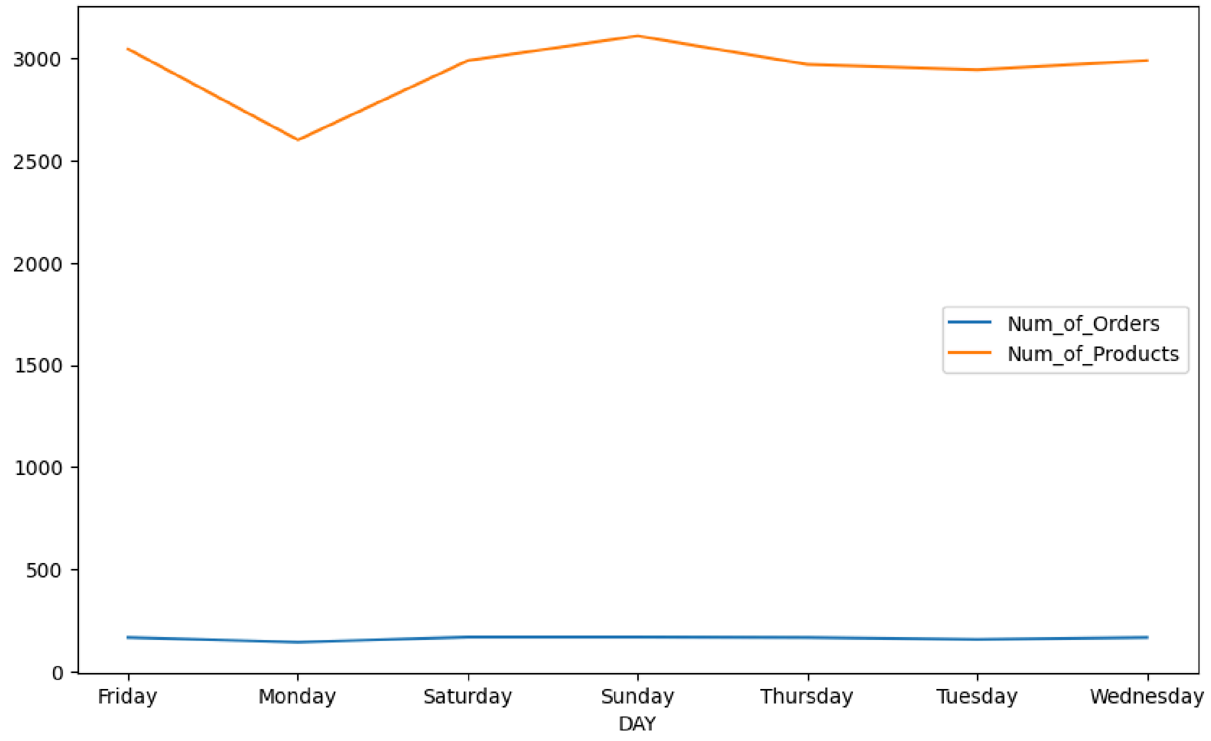
In [ ]:
```
df3 = df_datesplit.copy()
df_day = pd.DataFrame(df3[['DAY', 'Order_id', 'Product']].groupby('DAY').agg({'Orde
                                                                     'Product': 'count
newname = {'Order_id': 'Num_of_Orders', 'Product': 'Num_of_Products'}
df_day.rename(columns=newname, inplace=True)
df_day
```

Out[ ]:

	Num_of_Orders	Num_of_Products
DAY		
Friday	167	3044
Monday	143	2600
Saturday	169	2988
Sunday	169	3109
Thursday	167	2969
Tuesday	157	2943
Wednesday	167	2988

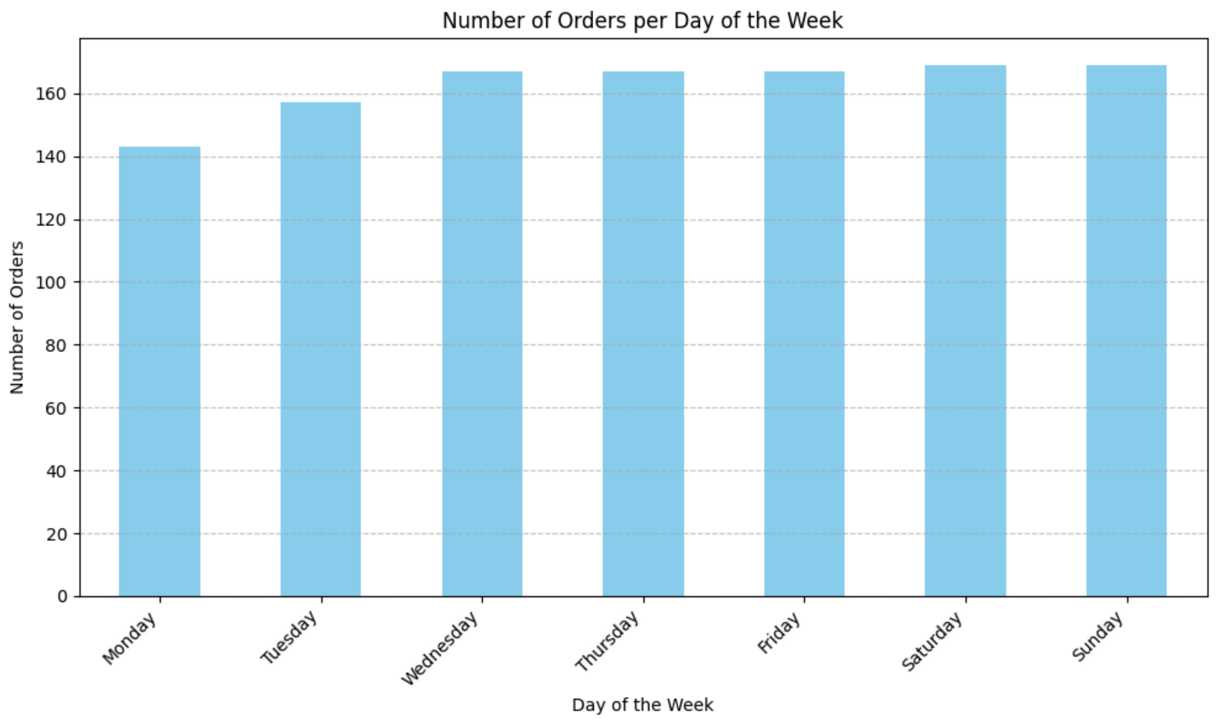
In [ ]:
```
#generate days of the week line chart for the Num_of_Orders and Num_of_Products
df_day.plot(figsize=(10, 6))
```

Out[ ]:
```
<Axes: xlabel='DAY'>
```

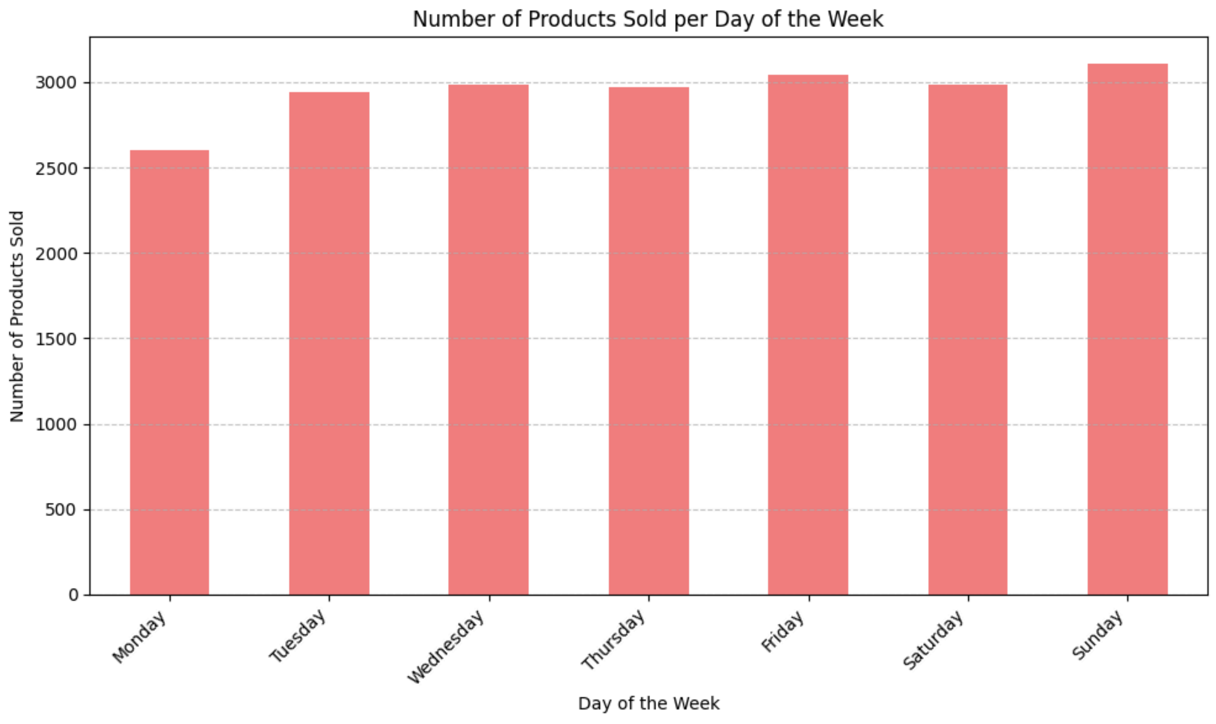


In [ ]:
```
!pip install mlxtend
```

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

plt.figure(figsize=(10, 6))
df_day_ordered['Num_of_Orders'].plot(kind='bar', color='skyblue')
plt.title('Number of Orders per Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Orders')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [ ]:
```python
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
df_day_ordered['Num_of_Products'].plot(kind='bar', color='lightcoral')
plt.title('Number of Products Sold per Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Products Sold')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [ ]:
```
# Read the data and convert Date to datetime
gro_df = pd.read_csv('/content/dataset_group.csv')
gro_df['Date'] = pd.to_datetime(gro_df['Date'], format='%d-%m-%Y')

# Copy the Date filed from gro_df and create date-related columns
df_datesplit = gro_df.copy()
df_datesplit['YEAR'] = pd.PeriodIndex(df_datesplit['Date'], freq='Y')
df_datesplit['MONTH'] = pd.PeriodIndex(df_datesplit['Date'], freq='M')
df_datesplit['QUARTER'] = pd.PeriodIndex(df_datesplit['Date'], freq='Q')
df_datesplit['WEEK'] = pd.PeriodIndex(df_datesplit['Date'], freq='W')
df_datesplit['DAY'] = df_datesplit.Date.dt.strftime('%A')
df_datesplit['MONTH_NAME'] = df_datesplit['Date'].dt.strftime('%b')

# Aggregate data for df_day
df3 = df_datesplit.copy()
df_day = pd.DataFrame(df3[['DAY', 'Order_id', 'Product']].groupby('DAY').agg({'Orde
                                                                  'Product': 'count
newname = {'Order_id': 'Num_of_Orders', 'Product': 'Num_of_Products'}
df_day.rename(columns=newname, inplace=True)

# 1. Define the custom order (starting Monday)
day_order = [
    'Monday', 'Tuesday', 'Wednesday', 'Thursday',
    'Friday', 'Saturday', 'Sunday'
]

# 2. Reindex df_day to sort by the custom day_order
df_day_ordered = df_day.reindex(day_order)

print("--- Data Preview (Now ordered Monday to Sunday) ---")
print(df_day_ordered)
```

```
--- Data Preview (Now ordered Monday to Sunday) ---
           Num_of_Orders  Num_of_Products
DAY
Monday               143             2600
Tuesday              157             2943
Wednesday            167             2988
Thursday             167             2969
Friday               167             3044
Saturday             169             2988
Sunday               169             3109
```

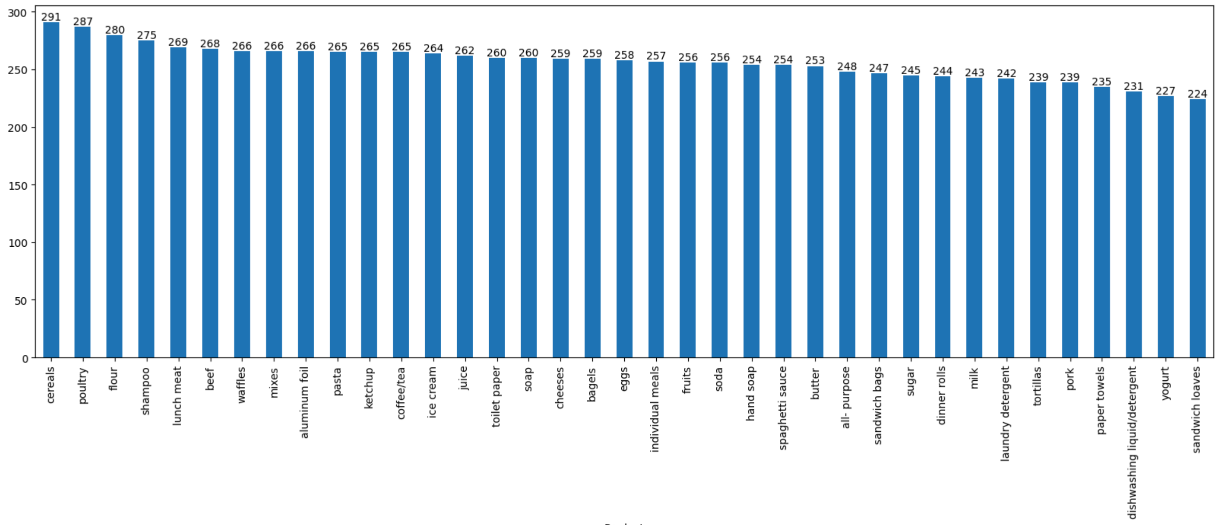
In [ ]:
```
# Count occurrences of each product
#filter products sold in 2018 from the gro_df
df_2018 = gro_df[gro_df['Date'].dt.year == 2018]

product_counts_2018 = df_2018['Product'].value_counts()

# Display the product counts
display(product_counts_2018)
```

	count
Product	
cereals	291
poultry	287
flour	280
shampoo	275
lunch meat	269
beef	268
waffles	266
mixes	266
aluminum foil	266
pasta	265
ketchup	265
coffee/tea	265
ice cream	264
juice	262
toilet paper	260
soap	260
cheeses	259
bagels	259
eggs	258
individual meals	257
fruits	256
soda	256
hand soap	254
spaghetti sauce	254
butter	253
all- purpose	248
sandwich bags	247
sugar	245
dinner rolls	244

Product	count
milk	243
laundry detergent	242
tortillas	239
pork	239
paper towels	235
dishwashing liquid/detergent	231
yogurt	227
sandwich loaves	224

dtype: int64

```
In [ ]: #Top ordered products- 2018
        product_counts_2018.plot(kind='bar', figsize=(20, 6))

        #label each bar with the total count
        for index, value in enumerate(product_counts_2018):
            plt.text(index,
                     value,
                     str(value),
                     ha='center',
                     va='bottom')
```



```
In [ ]: # Count occurrences of each product
        #filter products sold in 2019 from the gro_df
        df_2019 = gro_df[gro_df['Date'].dt.year == 2019]

        product_counts_2019 = df_2019['Product'].value_counts()
```

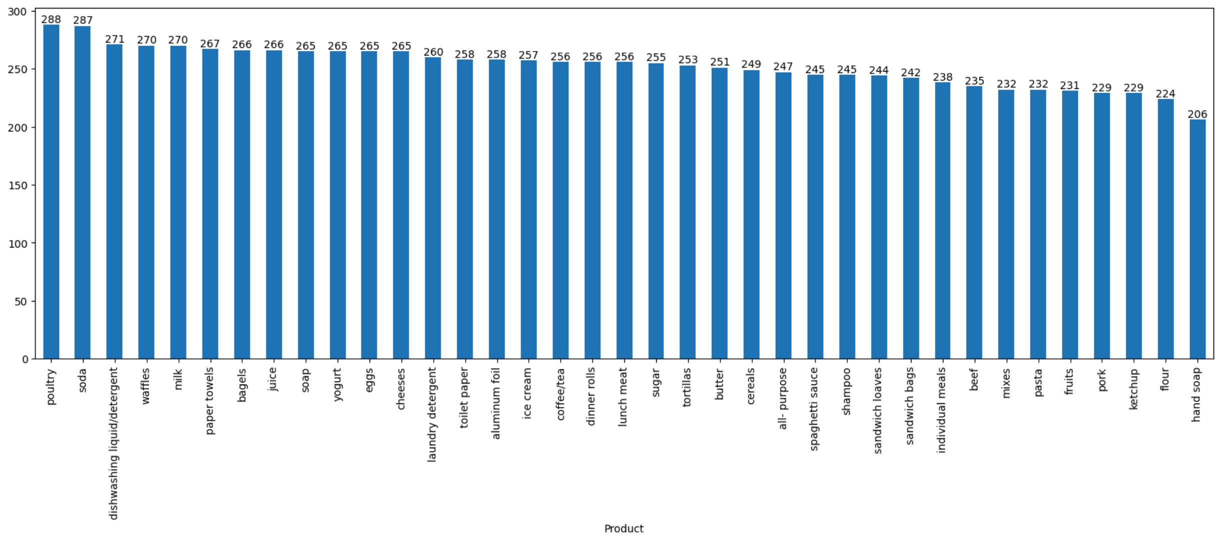
```
# Display the product counts
display(product_counts_2019)
```

	count
Product	
poultry	288
soda	287
dishwashing liquid/detergent	271
waffles	270
milk	270
paper towels	267
bagels	266
juice	266
soap	265
yogurt	265
eggs	265
cheeses	265
laundry detergent	260
toilet paper	258
aluminum foil	258
ice cream	257
coffee/tea	256
dinner rolls	256
lunch meat	256
sugar	255
tortillas	253
butter	251
cereals	249
all- purpose	247
spaghetti sauce	245
shampoo	245
sandwich loaves	244
sandwich bags	242
individual meals	238

	count
Product	
beef	235
mixes	232
pasta	232
fruits	231
pork	229
ketchup	229
flour	224
hand soap	206

dtype: int64

In [ ]:
```
#Top ordered products- 2019
product_counts_2019.plot(kind='bar', figsize=(20, 6))

#label each bar with the total count
for index, value in enumerate(product_counts_2019):
    plt.text(index,
             value,
             str(value),
             ha='center',
             va='bottom')
```



In [ ]:
```
# Count occurrences of each product
#filter products sold in 2020 from the gro_df
df_2020 = gro_df[gro_df['Date'].dt.year == 2020]

product_counts_2020 = df_2020['Product'].value_counts()
```

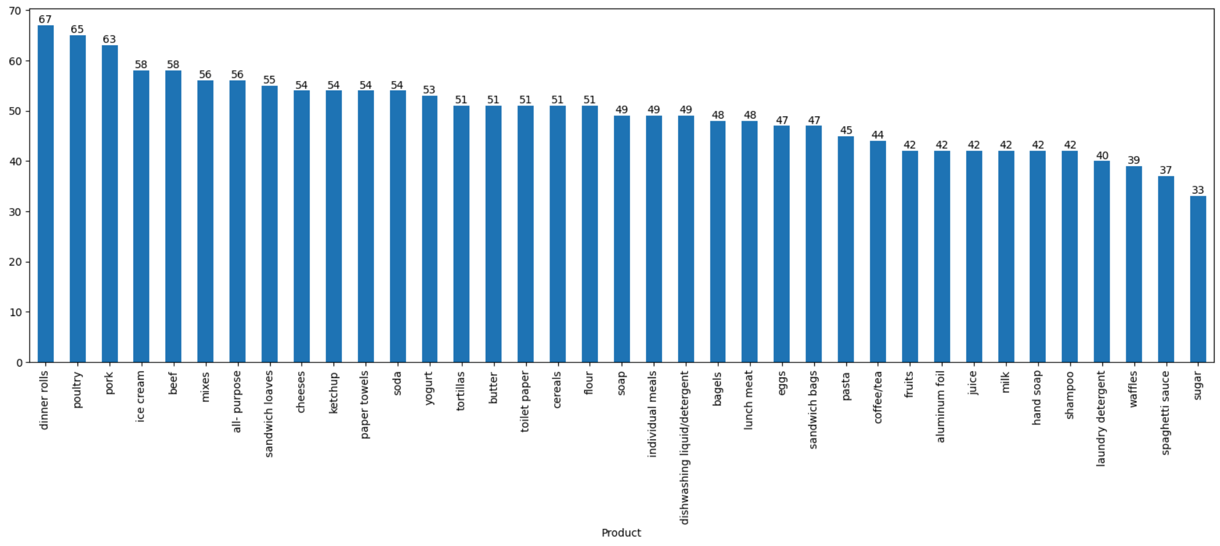
```
# Display the product counts
display(product_counts_2020)
```

	count
Product	
dinner rolls	67
poultry	65
pork	63
ice cream	58
beef	58
mixes	56
all- purpose	56
sandwich loaves	55
cheeses	54
ketchup	54
paper towels	54
soda	54
yogurt	53
tortillas	51
butter	51
toilet paper	51
cereals	51
flour	51
soap	49
individual meals	49
dishwashing liquid/detergent	49
bagels	48
lunch meat	48
eggs	47
sandwich bags	47
pasta	45
coffee/tea	44
fruits	42
aluminum foil	42

Product	count
juice	42
milk	42
hand soap	42
shampoo	42
laundry detergent	40
waffles	39
spaghetti sauce	37
sugar	33

dtype: int64

```
In [ ]: #Top ordered products- 2020
        product_counts_2020.plot(kind='bar', figsize=(20, 6))

        #label each bar with the total count
        for index, value in enumerate(product_counts_2020):
            plt.text(index,
                     value,
                     str(value),
                     ha='center',
                     va='bottom')
```


dinner rolls
poultry
pork
ice cream
beef
mixes
all- purpose
sandwich loaves
cheeses
ketchup
paper towels
soda
yogurt
tortillas
butter
toilet paper
cereals
flour
soap
individual meals
dishwashing liquid/detergent
bagels
lunch meat
eggs
sandwich bags
pasta
coffee/tea
fruits
aluminum foil
juice
milk
hand soap
shampoo
laundry detergent
waffles
spaghetti sauce
sugar
Product