顧客購買行為分析與行銷策略制定專案_2023年超商銷售資料集

數據來源: Kaggle-https://www.kaggle.com/datasets/hunter0007/ecommerce-dataset-for-predictive-marketing-2023/data

概述

在數位化時代，企業的決策模式正從以往倚賴經驗與直覺，逐步轉向依賴數據與分析結果。相較於傳統經驗導向的方式，數據驅動的決策不僅能提高精準度，更能即時掌握市場變化與消費者行為，協助企業在競爭激烈的環境中做出快速且有依據的應變。隨著資料蒐集、儲存與分析技術的日益成熟，越能掌握並有效運用數據的企業，往往能在產品優化、顧客經營與資源配置上展現出更高的效率與利潤潛力。

本模擬專案旨在透過分析顧客的實際購買紀錄與產品屬性資料，挖掘出其中潛在的消費模式與商品間的關聯性，進而提出具體可行的行銷策略建議。透過購物籃分析等資料探勘技術，本研究希望模擬企業在數據資源有限的情境下，如何運用基礎銷售資料進行有系統的分析，支援行銷決策，並為未來導入更全面的資料整合與客戶洞察奠定基礎。

專案架構

1. 建立環境與數據匯入
2. 數據清理與預處理
3. 探索性資料分析(EDA)-單變量分析與多變量分析
4. 關聯規則分析與Apriori
5. 行銷策略與執行
6. 限制

1. 建立環境與數據匯入

In [45]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [46]:
```
df = pd.read_csv('ECommerce_consumer behaviour.csv')
```

In [47]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19999 entries, 0 to 19998
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   order_id                19999 non-null  int64  
 1   user_id                 19999 non-null  int64  
 2   order_number            19999 non-null  int64  
 3   order_dow               19999 non-null  int64  
 4   order_hour_of_day       19999 non-null  int64  
 5   days_since_prior_order  18635 non-null  float64
 6   product_id              19999 non-null  int64  
 7   add_to_cart_order       19999 non-null  int64  
 8   reordered               19999 non-null  int64  
 9   department_id           19999 non-null  int64  
 10  department              19999 non-null  object 
 11  product_name            19999 non-null  object 
dtypes: float64(1), int64(9), object(2)
memory usage: 1.8+ MB
```

In [48]:
```
df.shape
```

Out[48]:
```
(19999, 12)
```

In [49]:
```
df.head(3)
```

Out[49]:

	order_id	user_id	order_number	order_dow	order_hour_of_day	days_since_prior_order	product_id	add_to_cart_order	reordered	department_id	department	product_
0	2425083	49125	1	2	18	NaN	17	1	0	13	pantry	b ingred
1	2425083	49125	1	2	18	NaN	91	2	0	16	dairy eggs	soy lactos
2	2425083	49125	1	2	18	NaN	36	3	0	16	dairy eggs	b

本數據集共有19,999筆資料及12個變項，其中會進一步分析的變項為:

1. order_dow 表示訂單在星期幾下單
2. order_hour_of_day 表示訂單下單的時間
3. add_to_cart_order 表示加入購物車的商品數量
4. reordered 是否重新下單
5. department 表示部門名稱
6. product_name 表示產品名稱

2. 數據清理與預處理

```
In [52]: #將星期轉為類別變項
df['order_dow'] = df['order_dow'].astype('category')
```

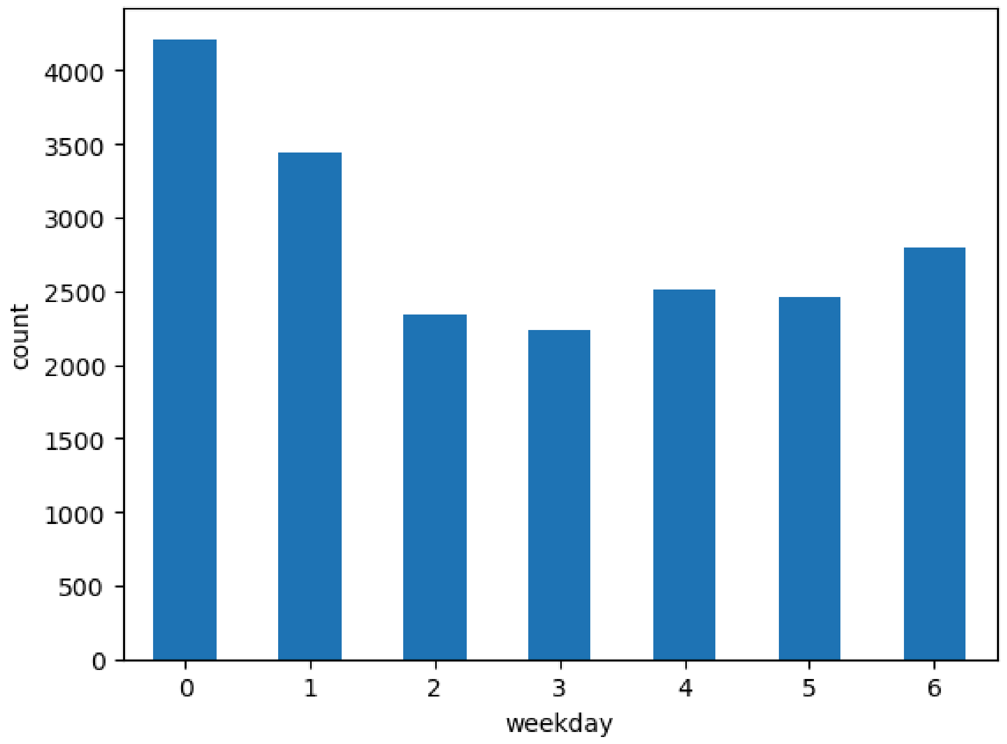
```
In [53]: #將NaN填入0
df['days_since_prior_order'] = df['days_since_prior_order'].fillna(0)
df['days_since_prior_order'].unique()
```

```
Out[53]: array([ 0.,  3.,  6.,  7., 30., 20.,  4.,  8., 15., 10., 28.,  9., 12.,
       11.,  2., 25., 13., 29., 14., 21.,  5.,  1., 18., 19., 17., 22.,
       26., 24., 16., 23., 27.])
```

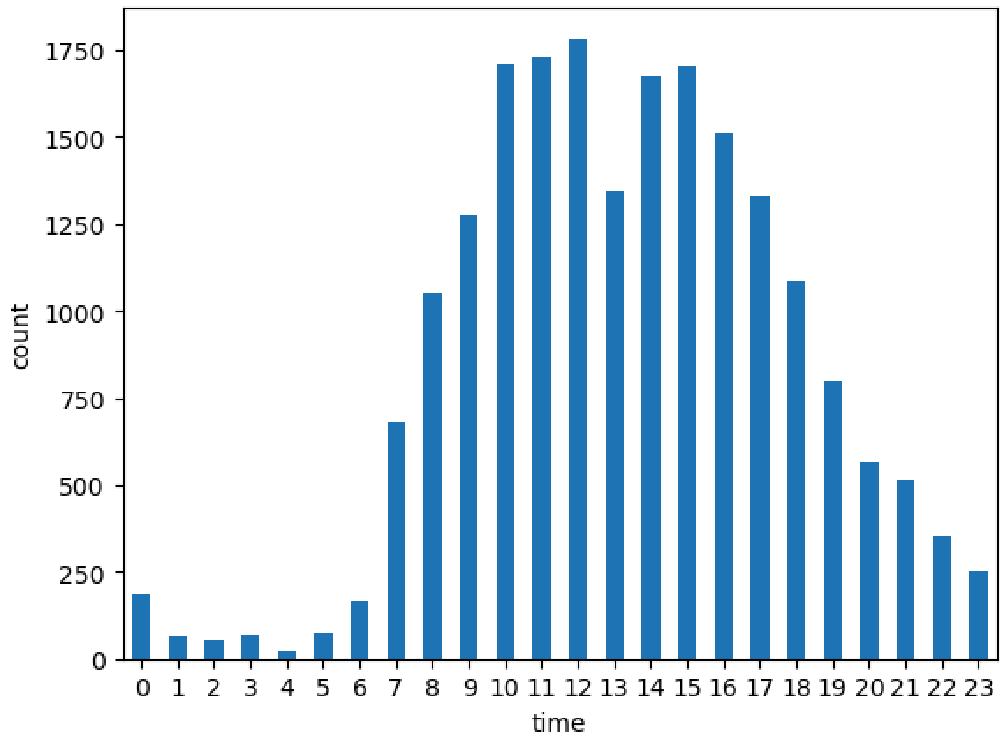
3. 探索性資料分析(EDA)

3.1 單變量分析

```
In [56]: df['order_dow'].value_counts().sort_index().plot(kind='bar', rot=0)
plt.xlabel('weekday')
plt.ylabel('count')
plt.show()
```

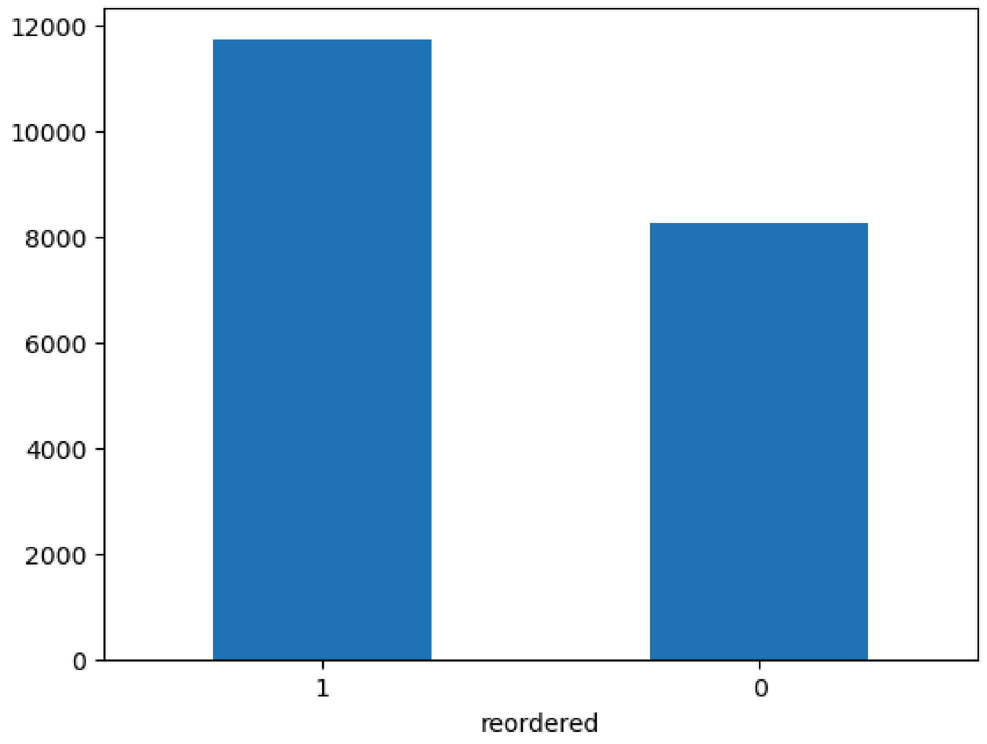


```
In [57]: df['order_hour_of_day'].value_counts().sort_index().plot(kind='bar', rot=0)
         plt.xlabel('time')
         plt.ylabel('count')
         plt.show()
```



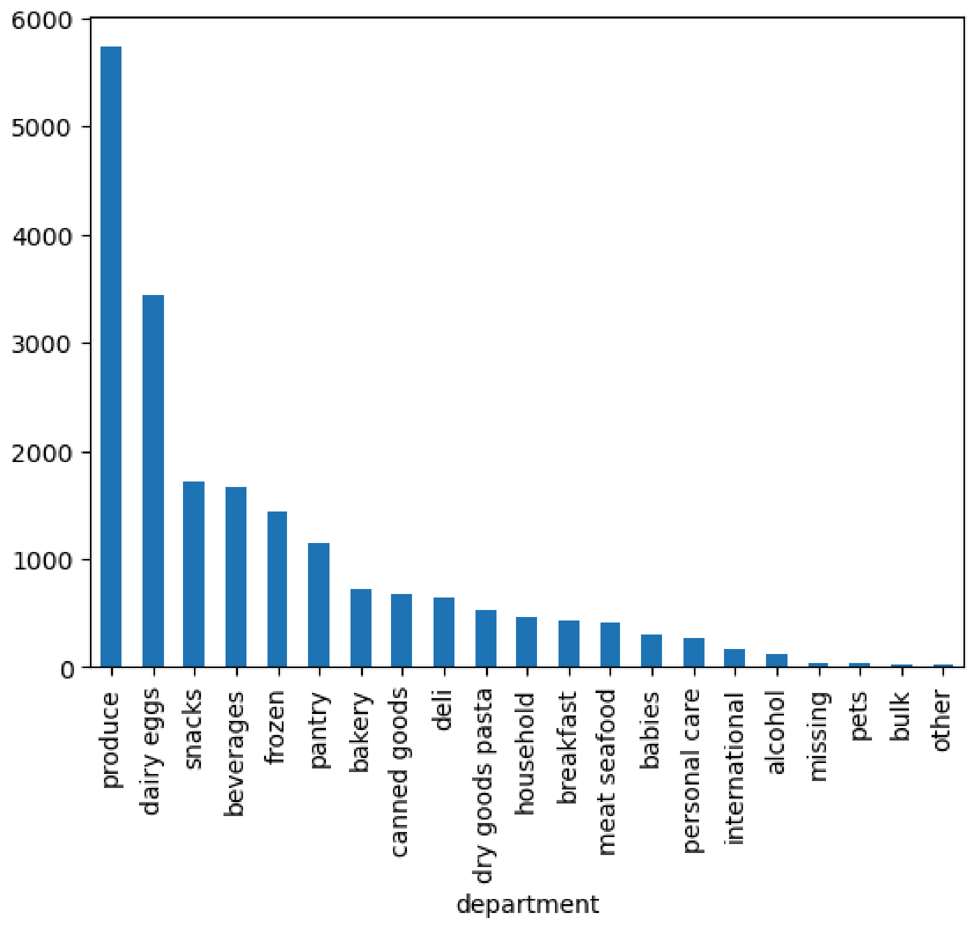
```
In [58]: df['reordered'].value_counts().plot(kind='bar', rot=0)
```

```
Out[58]: <Axes: xlabel='reordered'>
```



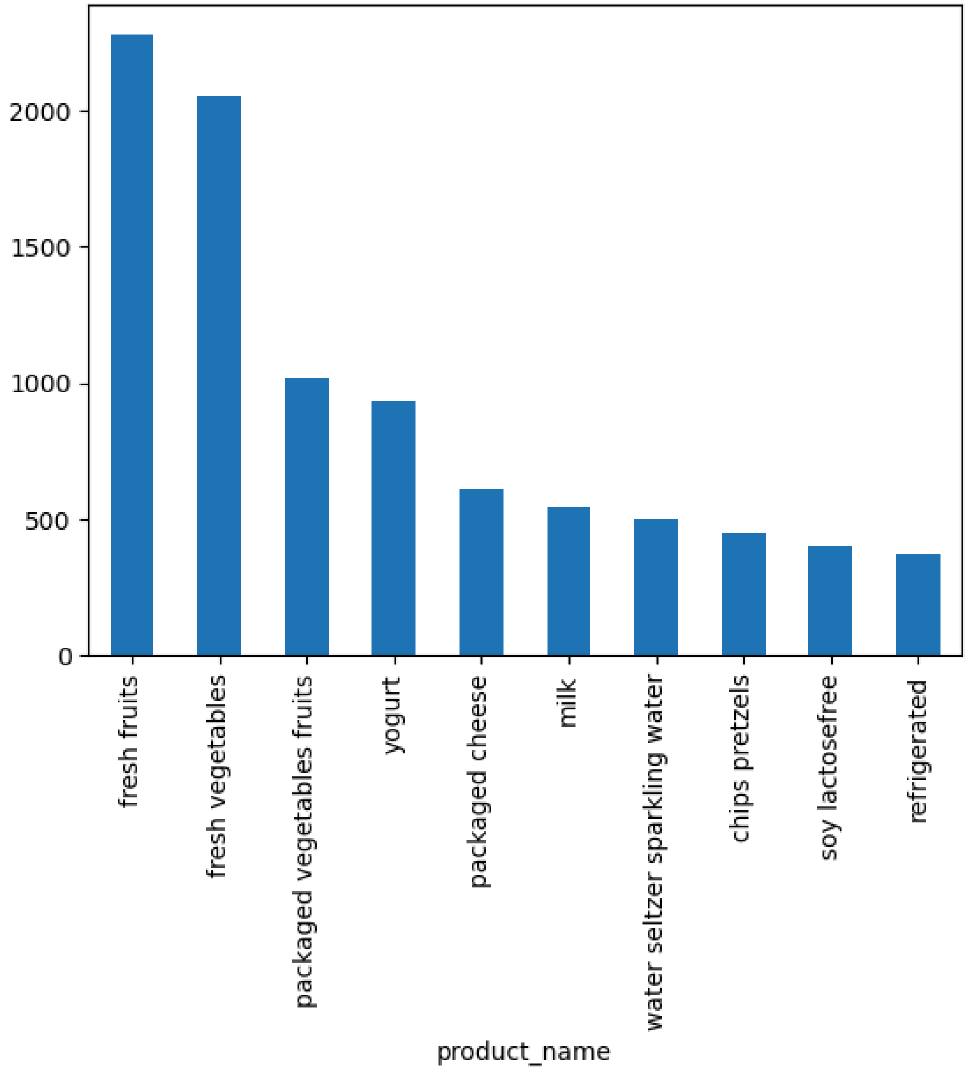
```
In [59]: df['department'].value_counts().plot(kind='bar', rot=90)
```

```
Out[59]: <Axes: xlabel='department'>
```



```
In [60]: df['product_name'].value_counts().head(10).plot(kind='bar', rot=90)
```

```
Out[60]: <Axes: xlabel='product_name'>
```



由以上視覺化圖表可知

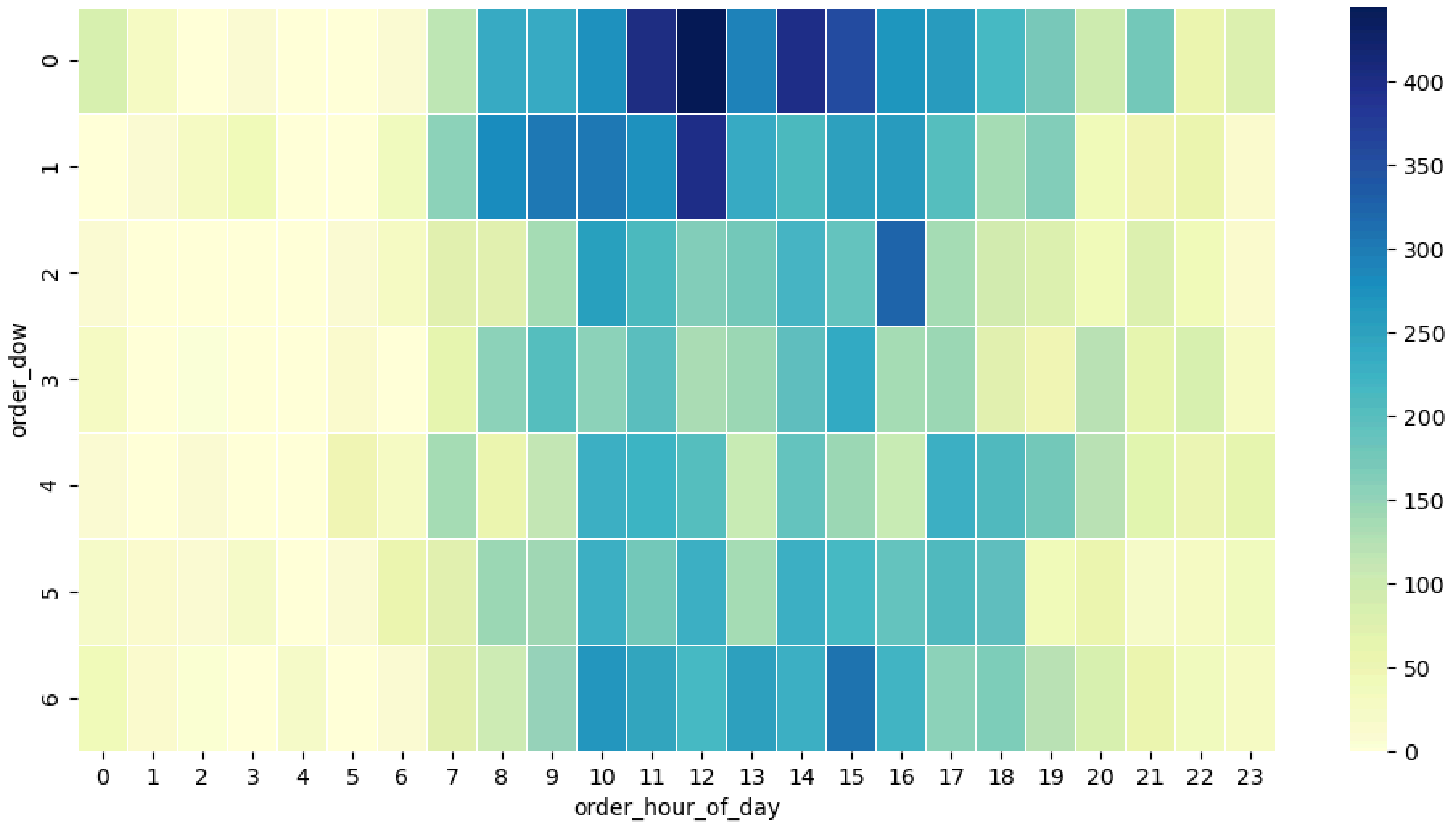
1. 大部分顧客消費時間落在周日、周一、周六的早上七點至晚上七點。
2. 超過一半的顧客會重購產品。
3. 生產、乳製品雞蛋、零食、飲料、冷凍，以上五個部門銷量最高。
4. 新鮮水果、新鮮蔬菜、已包裝蔬果、優格、已包裝起司為前五名銷量最佳產品。

3.2 多變量分析

In [63]:
```python
#星期與時間
heatmap_data = df.pivot_table(
    index='order_dow',
    columns='order_hour_of_day',
    values='order_id',
    aggfunc='count'  # 計算每個 (day, hour) 出現的訂單數
)
plt.figure(figsize=(12, 6))
sns.heatmap(heatmap_data, cmap='YlGnBu', linewidths=0.5)
```

```
C:\Users\linyi\AppData\Local\Temp\ipykernel_135212\2658438941.py:2: FutureWarning: The default value of observed=False is deprecated and will change to observed=T
rue in a future version of pandas. Specify observed=False to silence this warning and retain the current behavior
  heatmap_data = df.pivot_table(
```

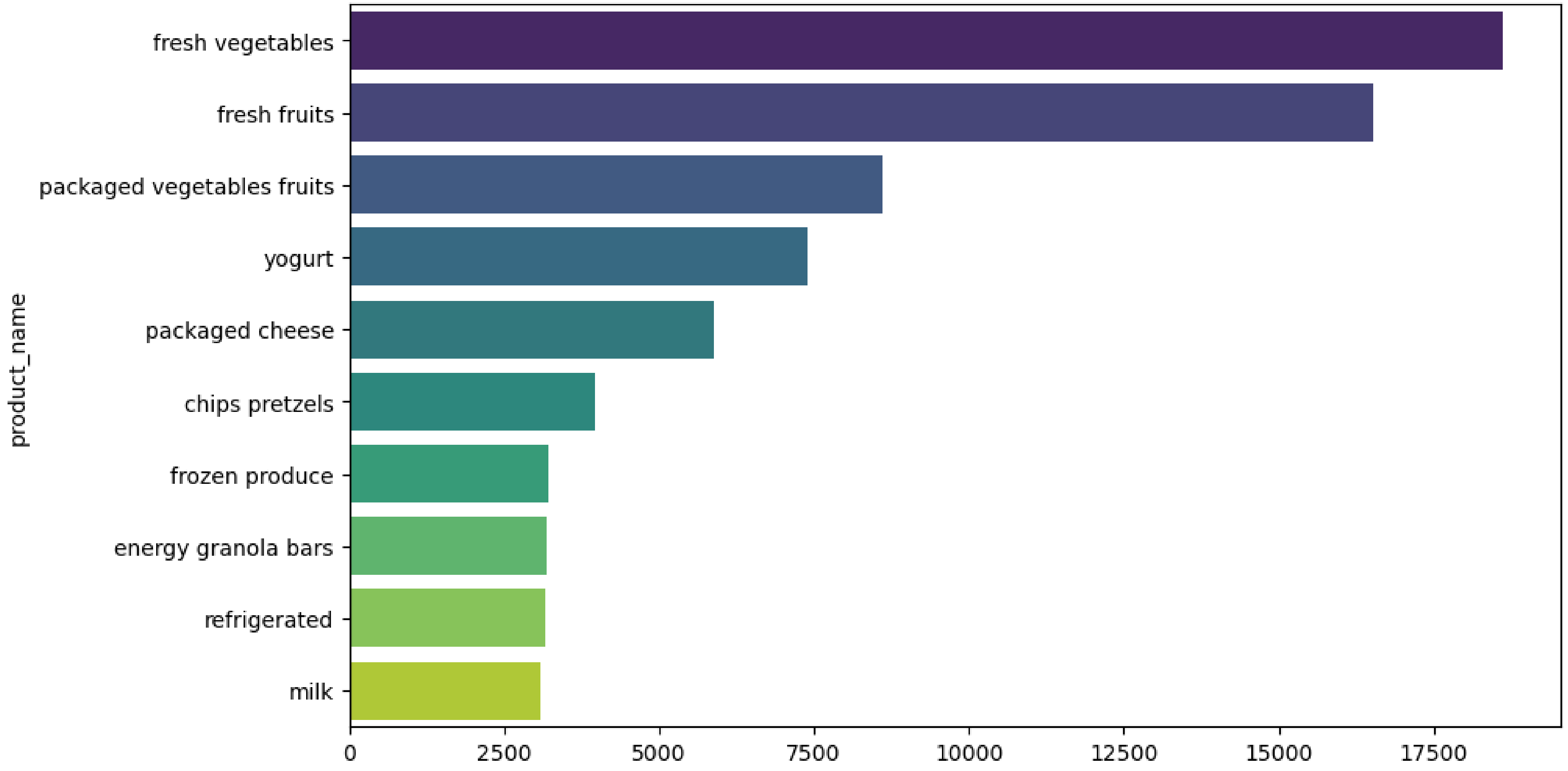
Out[63]:
```
<Axes: xlabel='order_hour_of_day', ylabel='order_dow'>
```



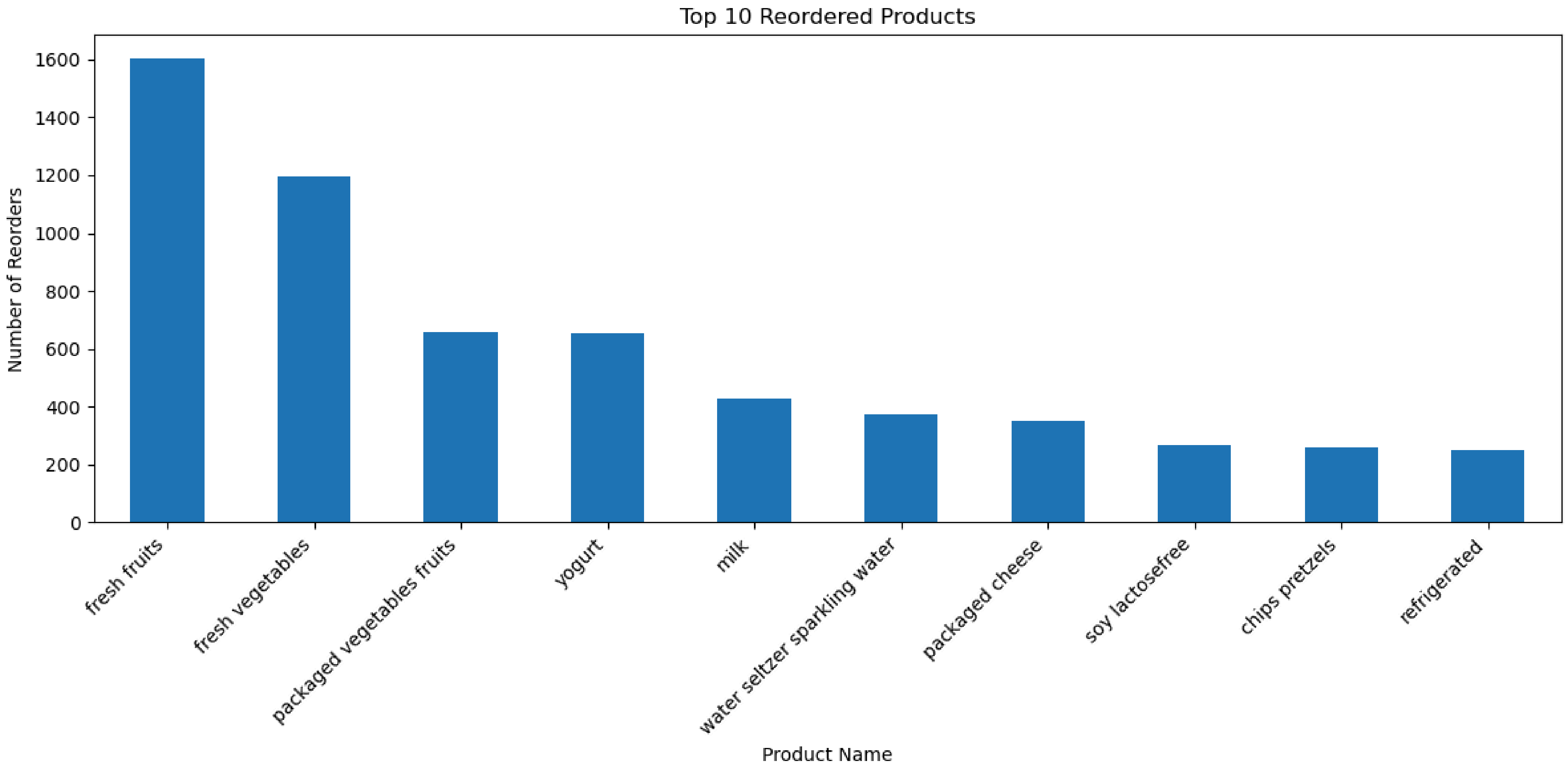
In [64]:
```python
#最多加入購物車的產品
top_products = (
    df.groupby('product_name')['add_to_cart_order']
    .sum()
    .sort_values(ascending=False)
    .head(10)
)

# 繪製條形圖
plt.figure(figsize=(10, 6))
sns.barplot(x=top_products.values, y=top_products.index, palette='viridis')
```

```
C:\Users\linyi\AppData\Local\Temp\ipykernel_135212\2494031713.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effec
t.

  sns.barplot(x=top_products.values, y=top_products.index, palette='viridis')
```

Out[64]:
```
<Axes: ylabel='product_name'>
```

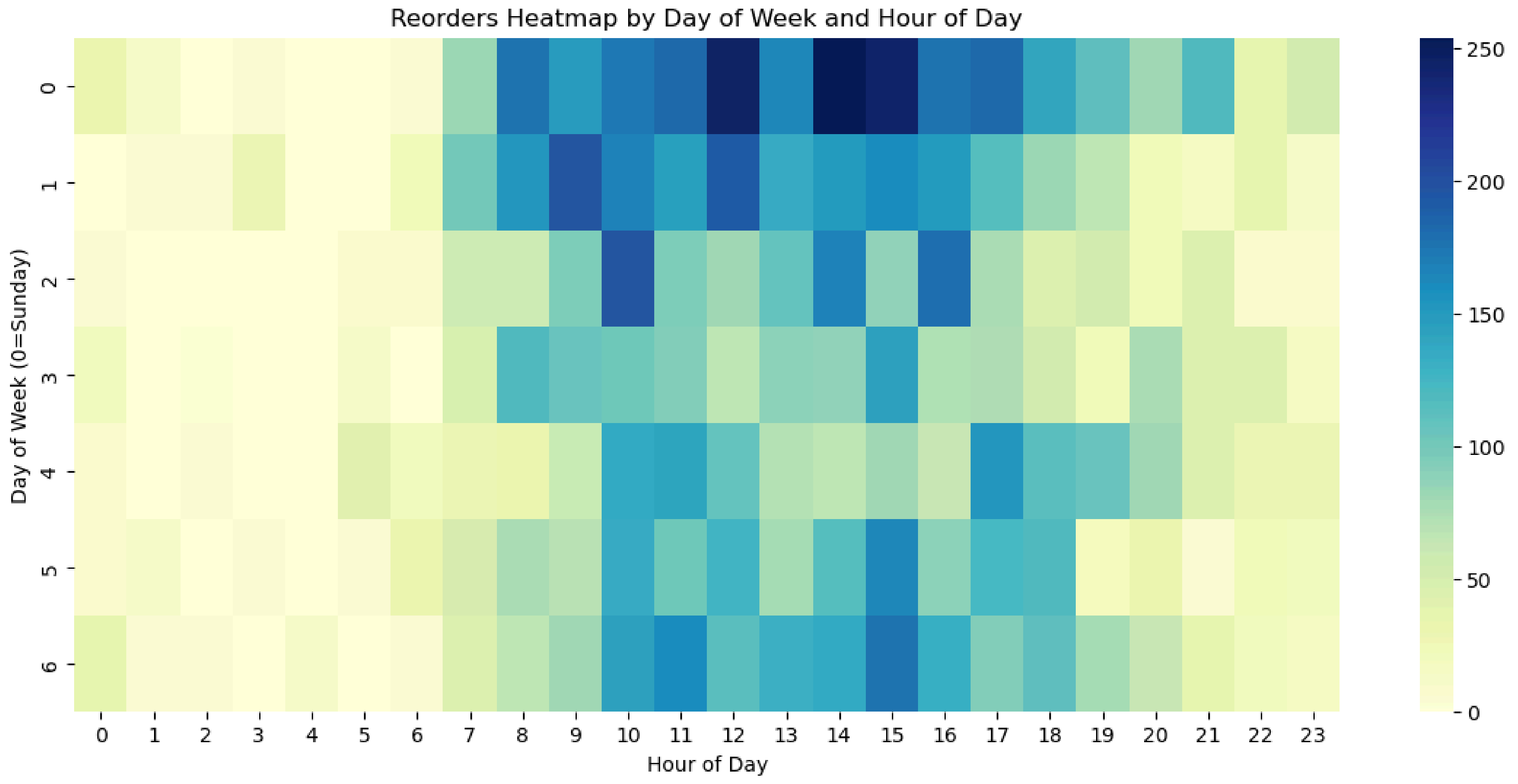


In [65]:
```
top_products = (
    df[df['reordered'] == 1]['product_name']
    .value_counts()
    .head(10)
)

# 繪圖
plt.figure(figsize=(12, 6))  # 避免太寬
top_products.plot(kind='bar')
plt.title('Top 10 Reordered Products')
plt.xlabel('Product Name')
plt.ylabel('Number of Reorders')
plt.xticks(rotation=45, ha='right')  # 讓標籤不重疊
plt.tight_layout()
plt.show()
```



In [66]:
```
#時段 + 回購率
heatmap_data = pd.pivot_table(
    df[df['reordered'] == 1],
    index='order_dow',
    columns='order_hour_of_day',
    values='reordered',
    aggfunc='count',
    fill_value=0
)

plt.figure(figsize=(14, 6))
sns.heatmap(heatmap_data, cmap='YlGnBu')
plt.title('Reorders Heatmap by Day of Week and Hour of Day')
plt.xlabel('Hour of Day')
plt.ylabel('Day of Week (0=Sunday)')
plt.show()
```

```
C:\Users\linyi\AppData\Local\Temp\ipykernel_135212\618937391.py:2: FutureWarning: The default value of observed=False is deprecated and will change to observed=Tr
ue in a future version of pandas. Specify observed=False to silence this warning and retain the current behavior
  heatmap_data = pd.pivot_table(
```



In [67]:
```
# AGGREGATING & GROUPING VALUES TO UNDERSTAND PURCHASING BEHAVIOUR
grouped = df.groupby(["product_id","product_name","department"])["reordered"].aggregate('count').reset_index()
grouped = grouped.sort_values(by='reordered', ascending=False)[:15].reset_index()
grouped
```

Out[67]:

	index	product_id	product_name	department	reordered
0	23	24	fresh fruits	produce	2276
1	82	83	fresh vegetables	produce	2055
2	121	123	packaged vegetables fruits	produce	1015
3	118	120	yogurt	dairy eggs	936
4	20	21	packaged cheese	dairy eggs	611
5	83	84	milk	dairy eggs	549
6	113	115	water seltzer sparkling water	beverages	502
7	106	107	chips pretzels	snacks	446
8	90	91	soy lactosefree	dairy eggs	405
9	30	31	refrigerated	beverages	370
10	111	112	bread	bakery	368
11	114	116	frozen produce	frozen	333
12	36	37	ice cream ice	frozen	317
13	85	86	eggs	dairy eggs	290
14	77	78	crackers	snacks	276

由以上圖表可知

1. 下單時間與重購時間大多集中在週日的上午十一點到下午三點，其次為周一的早上八點至下午四點與周六上午十點到下午四點。
2. 加入購物車最多的產品是新鮮蔬菜，超過17,500筆，其次為新鮮水果、已包裝蔬果、優格。
3. 重購次數最高者為新鮮水果、新鮮蔬菜、已包裝蔬果、優格。

4. 顧客行為分析

使用Apriori與關聯規則分析找出哪些商品經常被一起購買

In [74]:
```
!pip install mlxtend
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

```python
# 將每筆訂單的產品轉為列表
transactions = df.groupby("order_id")['product_name'].apply(list).values.tolist()

# 編碼為 one-hot 布林矩陣
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df_encoded = pd.DataFrame(te_ary, columns=te.columns_)

# Apriori分析
frequent_itemsets = apriori(df_encoded, min_support=0.01, use_colnames=True)

# 關聯規則分析
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1.0)

print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Collecting mlxtend
  Downloading mlxtend-0.23.4-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in c:\users\linyi\anaconda3\lib\site-packages (from mlxtend) (1.13.1)
Requirement already satisfied: numpy>=1.16.2 in c:\users\linyi\anaconda3\lib\site-packages (from mlxtend) (1.26.4)
Requirement already satisfied: pandas>=0.24.2 in c:\users\linyi\anaconda3\lib\site-packages (from mlxtend) (2.2.2)
Requirement already satisfied: scikit-learn>=1.3.1 in c:\users\linyi\anaconda3\lib\site-packages (from mlxtend) (1.6.1)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\linyi\anaconda3\lib\site-packages (from mlxtend) (3.8.4)
Requirement already satisfied: joblib>=0.13.2 in c:\users\linyi\anaconda3\lib\site-packages (from mlxtend) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\linyi\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\linyi\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\linyi\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\linyi\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\linyi\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\linyi\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\linyi\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\linyi\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\linyi\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\linyi\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2023.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\linyi\anaconda3\lib\site-packages (from scikit-learn>=1.3.1->mlxtend) (3.6.0)
Requirement already satisfied: six>=1.5 in c:\users\linyi\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)
Downloading mlxtend-0.23.4-py3-none-any.whl (1.4 MB)
   ---------------------------------------- 0.0/1.4 MB ? eta -:--:--
   ---------------------------------------- 1.4/1.4 MB 23.7 MB/s eta 0:00:00
Installing collected packages: mlxtend
Successfully installed mlxtend-0.23.4
                        antecedents  \
0                    (fresh fruits)
1                     (asian foods)
2                     (asian foods)
3                (fresh vegetables)
4                     (asian foods)
...                             ...
30961                (refrigerated)
30962  (packaged vegetables fruits)
30963            (fresh vegetables)
30964             (packaged cheese)
30965                (fresh fruits)

                                             consequents   support  \
0                                          (asian foods)  0.027514
1                                         (fresh fruits)  0.027514
2                                     (fresh vegetables)  0.027514
3                                          (asian foods)  0.027514
4                                                 (milk)  0.013507
...                                                  ...       ...
30961  (yogurt, packaged vegetables fruits, packaged ...  0.010005
30962  (yogurt, refrigerated, packaged cheese, fresh ...  0.010005
30963  (yogurt, refrigerated, packaged vegetables fru...  0.010005
30964  (yogurt, refrigerated, packaged vegetables fru...  0.010005
30965  (yogurt, refrigerated, packaged vegetables fru...  0.010005
       confidence      lift
0        0.049639  1.102537
1        0.611111  1.102537
2        0.611111  1.405767
3        0.063291  1.405767
4        0.300000  1.236495
...           ...       ...
30961    0.074627  2.043549
30962    0.028777  2.212507
30963    0.023015  2.091223
30964    0.044743  2.630609
30965    0.018051  1.718240

[30966 rows x 5 columns]
```

In [76]:
```
# 篩選出有意義的規則，再取前 10 筆
filtered_rules = rules[
    (rules['support'] >= 0.01) &
    (rules['confidence'] >= 0.5) &
    (rules['lift'] >= 1.2)
]

top_filtered = filtered_rules.sort_values(by='lift', ascending=False).head(10)
print(top_filtered[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
                                           antecedents  \
30429  (yogurt, packaged vegetables fruits, fresh herbs)
11223      (packaged vegetables fruits, bread, crackers)
21536                 (fresh herbs, other creams cheeses)
23045          (baking ingredients, yogurt, fresh fruits)
27343                               (yogurt, fresh herbs)
30384       (fresh dips tapenades, milk, fresh vegetables)
30416  (fresh fruits, yogurt, packaged vegetables fru...
27335                 (fresh fruits, yogurt, fresh herbs)
30852  (fresh fruits, yogurt, fresh vegetables, other...
23053                        (baking ingredients, yogurt)

                                           consequents   support  \
30429  (fresh fruits, packaged cheese, fresh vegetables)  0.010005
11223                                       (lunch meat)  0.010005
21536                (packaged cheese, fresh vegetables)  0.010505
23045                (packaged cheese, fresh vegetables)  0.013507
27343  (fresh fruits, packaged cheese, fresh vegetables)  0.014507
30384  (fresh fruits, yogurt, packaged vegetables fru...  0.010505
30416                (packaged cheese, fresh vegetables)  0.010005
27335                (packaged cheese, fresh vegetables)  0.014507
30852      (packaged vegetables fruits, packaged cheese)  0.011506
23053  (fresh fruits, packaged cheese, fresh vegetables)  0.013507

       confidence      lift
30429    0.555556  5.497800
11223    0.588235  5.369326
21536    0.677419  5.352416
23045    0.675000  5.333300
27343    0.537037  5.314540
30384    0.552632  5.162199
30416    0.645161  5.097539
27335    0.644444  5.091875
30852    0.522727  5.023711
23053    0.500000  4.948020
```

根據以上分析結果，將顧客分為以下三群:

1. 健康飲食族:同時購買新鮮蔬果、優格、香料、起司
2. 自備便當、三明治、沙拉族群:同時購買冷盤肉、餅乾、麵包、蔬果
3. 烘焙族群:同時購買烘焙原料與乳製品

5. 行銷策略與執行

綜合以上分析結果，制定以下行銷策略:

1. 推出會員專屬優惠：針對重購次數最高的產品（如新鮮水果、蔬菜、優格），可提供「買N送1」或積點兌換機制。
2. 精準時段推播優惠通知：每週週日11點–15點、週一8點–16點、週六10點–16點，透過App推播或Email發送限時優惠券，促進即時轉換。

3. 限時搶購活動：利用熱門時段舉辦「兩小時限時買一送一」活動，特別是針對保存期限較短的乳製品。
4. 針對健康飲食族推廣營養食譜、營養知識。
5. 針對自備便當族群推出冷盤組合(麵包、肉盤、蔬菜、醬料...)、跨品類陳列產品，亦可推出購買組合包集可加價購環保餐具的策略。
6. 針對烘焙族群推出料理教學活動並推出報名即可以優惠價格購買食材。

6. 限制

本次購物籃分析主要聚焦於顧客的實際購買行為，透過 Apriori 演算法找出商品間的聯合購買關係，雖能揭露部分潛在的消費習慣，但整體分析仍存在以下幾點限制：

1. 缺乏顧客背景與外部因素資料，行銷策略流於片面

本專案所使用的資料僅涵蓋顧客的購買紀錄與商品資訊，並未涵蓋顧客的基本屬性（如年齡、性別、收入）、心理特徵（如價值觀、動機）、生活型態（如家庭結構、飲食偏好）與接觸管道（如是否受到行銷訊息影響），也無納入季節性、地區性、促銷活動等外部因素。因此，雖然分析結果揭示了某些商品之間的強關聯，但這些模式未必能準確對應特定客群或消費情境，導致行銷策略可能難以精準定位或有效執行，降低策略的全面性與實效性。

2. 缺乏財務資訊支撐，限制資源配置與策略可行性評估

分析結果雖可用於制定促銷或商品組合策略，但未整合商品的利潤率、庫存成本、行銷費用與營收貢獻等財務指標，因此在制定行銷或營運策略時，無法有效評估各項建議的資源投入回報比與實施優先順序。舉例來說，即使某商品組合具有高度關聯性，若其中商品邊際利潤過低或存貨週轉率差，貿然推行仍可能導致成本浪費或產能錯配。因此，缺乏財務面數據限制了策略的可行性與決策依據完整性。