In [1]:
```
# ============================
#  Part 1: 导入Pivot表数据
# ============================

import pandas as pd
import numpy as np
from pulp import LpProblem, LpMinimize, LpVariable, lpSum, LpStatus, valu

# 读取pivot表
df = pd.read_excel("pivot-tables.xlsx", sheet_name="Sheet1")

# 取月份
df["Month"] = pd.to_datetime(df["Date"]).dt.month

# 只保留水果（Apple, Banana, Mango, Orange）
fruits = ["Apple", "Banana", "Mango", "Orange"]
df_fruit = df[df["Product"].isin(fruits)]

# 生成透视表
pivot_df = pd.pivot_table(df_fruit,
                          index="Month",
                          columns="Product",
                          values="Amount",
                          aggfunc="sum",
                          fill_value=0)

pivot_df = pivot_df.reset_index()  # 添加Month列
pivot_df["Month_num"] = pivot_df["Month"]  # 保证Month_num有

print("Pivot Table:")
print(pivot_df)

# 计算全年总销售金额 Grand Total
total_sales = pivot_df[["Apple", "Banana", "Mango", "Orange"]].sum().sum(
print("\nGrand Total Sales:", total_sales)
```

```
Pivot Table:
Product  Month  Apple  Banana  Mango  Orange  Month_num
0            1  16794   29728      0    3610          1
1            2  19715    9228   9029    2256          2
2            3  25702   26224   3663   15869          3
3            4  14586   16001      0    1113          4
4            5  22557   69521  33384   23790          5
5            6   6126   15208      0    4514          6
6            7   2034   31336   5480   14548          7
7            8  22611    9980      0     859          8
8            9   8489   51835   5523   10048          9
9           10  15331   22320      0       0         10
10          11  11978   29530      0   24091         11
11          12  25334   29384      0    3740         12

Grand Total Sales: 693069
```

In [2]:
```
# ============================
#  Part 2: 导入价格数据 + 计算季节性趋势
# ============================

# 读取香蕉价格
banana_df = pd.read_excel("bananas-120.xlsx")
```

```
banana_df["Year"] = pd.to_datetime(banana_df["Month"]).dt.year
banana_df["Month"] = pd.to_datetime(banana_df["Month"]).dt.month
banana_df = banana_df[((banana_df["Year"] > 2016) | (banana_df["Month"] >
                       ((banana_df["Year"] < 2024) | (banana_df["Month"] <

# 读取橙子价格
orange_df = pd.read_excel("oranges-120.xlsx")
orange_df["Year"] = pd.to_datetime(orange_df["Month"]).dt.year
orange_df["Month"] = pd.to_datetime(orange_df["Month"]).dt.month
orange_df = orange_df[((orange_df["Year"] > 2016) | (orange_df["Month"] >
                       ((orange_df["Year"] < 2024) | (orange_df["Month"] <

# 读取CPI（Apple和Mango使用CPI）
cpi_df = pd.read_csv("cpi_data.csv")
cpi_df["Year"] = pd.to_datetime(cpi_df["DATE"]).dt.year
cpi_df["Month"] = pd.to_datetime(cpi_df["DATE"]).dt.month
cpi_df = cpi_df[(cpi_df["Year"] >= 2016) & (cpi_df["Year"] <= 2024)]

# 计算季节性趋势函数
def compute_seasonal_trend(df):
    result = []
    for m in range(1, 13):
        month_prices = []
        for year in df["Year"].unique():
            temp = df[(df["Year"] == year) & (df["Month"] == m)]
            jan = df[(df["Year"] == year) & (df["Month"] == 1)]
            if len(temp) == 0 or len(jan) == 0:
                continue
            price = temp["Price"].values[0]
            price_jan = jan["Price"].values[0]
            rescaled = price / price_jan
            month_prices.append(rescaled)
        if len(month_prices) > 0:
            result.append(sum(month_prices) / len(month_prices))
        else:
            result.append(1.0)
    return result

# 计算每个水果的季节性趋势
banana_price = compute_seasonal_trend(banana_df)
orange_price = compute_seasonal_trend(orange_df)
cpi_trend = compute_seasonal_trend(cpi_df)

apple_price = cpi_trend
mango_price = cpi_trend

# 构建价格字典
price_table = {
    "Apple": apple_price,
    "Banana": banana_price,
    "Mango": mango_price,
    "Orange": orange_price
}

print("\n价格季节性趋势 (Seasonal Trends):")
for fruit in price_table:
    print(f"{fruit}: {price_table[fruit]}")
```

```
价格季节性趋势 (Seasonal Trends):
Apple: [1.0, 1.0046501888331125, 1.009239948819114, 1.0129598984512989, 1.
0168592787049058, 1.021189432812703, 1.0225763841364672, 1.024210440799017
6, 1.0264854434679693, 1.0285916094804117, 1.028009015194604, 1.0276160461
552069]
Banana: [1.0, 1.0250763400781011, 1.0321841224836716, 1.0439060104525677,
1.027118503103171, 1.0041747699956374, 1.0033303512053244, 1.0134027078983
858, 1.0032142160975592, 0.9795584199085788, 0.9756524969198583, 1.0416297
97295998]
Mango: [1.0, 1.0046501888331125, 1.009239948819114, 1.0129598984512989, 1.
0168592787049058, 1.021189432812703, 1.0225763841364672, 1.024210440799017
6, 1.0264854434679693, 1.0285916094804117, 1.028009015194604, 1.0276160461
552069]
Orange: [1.0, 1.0214173573094767, 1.032469174191858, 1.0618490678871484,
1.1025078590693318, 1.1231429159953725, 1.153388230614327, 1.1769397902268
115, 1.215735261771088, 1.2472570913597003, 1.276269438378582, 1.207705645
7052273]
```

Let x_{ijk} represent the dollar amount spent in month i to purchase fruit type k for fulfilling the sales demand in month j, where the fruit index $k = 0, 1, 2, 3$ corresponds to apples, bananas, mangos, and oranges, respectively.
Purchases are allowed only for the current or future sales months, i.e., $i \leq j$.

The objective is to minimize the total purchasing cost, formulated as:

$$\text{Minimize} \quad Z = \sum_{k=0}^{3} \sum_{j=1}^{12} \sum_{i=1}^{j} x_{ijk}$$

$$\begin{aligned}
\text{subject to} \quad & x_{ijk} \text{ denotes the dollar amount spent in month } i \\
& \quad \text{on fruit type } k \text{ to (partially or fully) satisfy the sales in month } j, \\
& i, j \in \{1, 2, \ldots, 12\}, \\
& k \in \{0, 1, 2, 3\}.
\end{aligned}$$

The following constraints ensure that the sales demand in month j is fully covered by advance purchases from months 1 to j:

$$\sum_{i=1}^{j} x_{ijk} \cdot \frac{p_{jk}}{p_{ik}} \geq s_{jk} \quad \forall\, j = 1, \ldots, 12, \quad k = 0, 1, 2, 3.$$

Let x_{ijk} represent the dollar amount spent in month i to purchase fruit type k for fulfilling the sales demand in month j.
Let GT denote the "Grand Total" of all fruits for the year obtained from the pivot table.

The following quarterly cash-flow constraints ensure that the total spending in each quarter does not exceed 30% of the annual total:

$$\sum_{k=0}^{3} \sum_{i \in Q_q} \sum_{j=i}^{12} x_{ijk} \leq 0.3 \cdot \mathrm{GT} \quad \forall\, q = 1, 2, 3, 4$$

where the sets Q_q represent the months in each quarter:

$$Q_1 = \{1, 2, 3\}, \quad Q_2 = \{4, 5, 6\}, \quad Q_3 = \{7, 8, 9\}, \quad Q_4 = \{10, 11, 12\}.$$

In [3]:
```
# ============================
#  Part 3: 构建优化模型
# ============================

from pulp import LpProblem, LpMinimize, LpVariable, lpSum, LpStatus, valu
from datetime import datetime

model = LpProblem("Advanced_Purchasing", LpMinimize)

months = list(range(1, 13))
fruits = ["Apple", "Banana", "Mango", "Orange"]

# 决策变量 x[i, j, k]，仅定义 i <= j
x = {}
for k, fruit in enumerate(fruits):
    for j in months:
        for i in range(1, j + 1):
            x[(i, j, k)] = LpVariable(f"x_{i}_{j}_{k}", lowBound=0)

# ============================
#  Part 4: 添加目标与约束
# ============================

# 目标函数：最小化实际采购金额
model += lpSum(x[(i, j, k)] for (i, j, k) in x)

# 需求约束：每月销售额必须由提前采购折算后满足
for k, fruit in enumerate(fruits):
    for j in months:
        model += (
            lpSum(
                x[(i, j, k)]
                * (price_table[fruit][j-1] / price_table[fruit][i-1])
                for i in range(1, j + 1)
            ) >= pivot_df.loc[pivot_df["Month_num"] == j, fruit].values[0
            f"Sales_Constraint_{fruit}_{j}"
        )

# 季度现金流约束
quarters = {
    "Q1": [1, 2, 3],
    "Q2": [4, 5, 6],
    "Q3": [7, 8, 9],
    "Q4": [10, 11, 12]
}

for q, q_months in quarters.items():
    model += (
        lpSum(
            x[(i, j, k)]
            for k in range(4)
            for i in q_months
            for j in range(i, 13)
        ) <= 0.3 * total_sales,
        f"Cashflow_Constraint_{q}"
```

```
    )

# ============================
#  Part 5: 求解模型
# ============================

model.solve()
print("\nModel Status:", LpStatus[model.status])

# ============================
#  Part 6: 结果输出
# ============================

optimized_cost = value(model.objective)
saving_percentage = (total_sales - optimized_cost) / total_sales * 100

print(f"\n原始采购成本 (Baseline): ${total_sales:.2f}")
print(f"优化采购成本 (Optimal): ${optimized_cost:.2f}")
print(f"节省比例: {saving_percentage:.2f}%")

# ============================
#  Part 7: 采购计划整理与输出
# ============================

import pandas as pd

purchase_plan = pd.DataFrame(0.0, index=months, columns=fruits)

for k, fruit in enumerate(fruits):
    for i in months:
        purchase_plan.loc[i, fruit] = sum(
            value(x[(i, j, k)]) for j in range(i, 13)
        )

purchase_plan.index.name = "PurchaseMonth"

print("\n每月各水果采购金额:")
print(purchase_plan.round(2))

purchase_plan.to_excel("Purchasing_Plan.xlsx")
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /Users/liuliangjie/Library/Python/3.9/lib/python/site-packa
ges/pulp/apis/../solverdir/cbc/osx/i64/cbc /var/folders/vt/qwvfy06n63q184j
d56rxd3dh0000gn/T/9491aa9178bb4d0b8f8a15a23db2d7af-pulp.mps -timeMode elap
sed -branch -printingOptions all -solution /var/folders/vt/qwvfy06n63q184j
d56rxd3dh0000gn/T/9491aa9178bb4d0b8f8a15a23db2d7af-pulp.sol (default strat
egy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 57 COLUMNS
At line 994 RHS
At line 1047 BOUNDS
At line 1048 ENDATA
Problem MODEL has 52 rows, 312 columns and 624 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 41 (-11) rows, 247 (-65) columns and 494 (-130) elements
Perturbing problem by 0.001% of 1 - largest nonzero change 5.5339243e-05 (
0.0055339243%) - largest zero change 0
0  Obj 50132 Primal inf 642937 (37)
31  Obj 614512.48 Primal inf 93016.11 (15)
53  Obj 677334.76
Optimal - objective value 677309.26
After Postsolve, objective 677309.26, infeasibilities - dual 0 (0), primal
0 (0)
Optimal objective 677309.2558 - 53 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.01


Model Status: Optimal

原始采购成本 (Baseline): $693069.00
优化采购成本 (Optimal): $677309.25
节省比例: 2.27%

每月各水果采购金额:
                   Apple    Banana     Mango    Orange
PurchaseMonth
1              61884.44  64136.58  12616.67  69283.02
2                  0.00      0.00      0.00      0.00
3                  0.00      0.00      0.00      0.00
4              44983.72  16001.00  38706.21  23500.77
5                  0.00  69521.00      0.00      0.00
6                  0.00  15208.00      0.00      0.00
7              83563.24  41216.81   5480.00      0.00
8                  0.00      0.00      0.00      0.00
9                  0.00  51835.00      0.00      0.00
10                 0.00  22320.00      0.00      0.00
11                 0.00  57052.80      0.00      0.00
12                 0.00      0.00      0.00      0.00
```