In [1]:
```
##Step1
import pandas as pd

#Load data
df = pd.read_csv("C:/Users/sridh/Desktop/Personal/Projects/Growth Analyst II  Kerala Ayurveda Assignment/Amazon

#Rename columns for consistency
df.columns = df.columns.str.strip().str.replace(' ', '_').str.lower()

#Convert 'date' to datetime
df['date'] = pd.to_datetime(df['date'])

# Step 4: Clean columns with comma strings
cols_to_clean = [
    'total_sales', 'organic_sales', 'ad_sales',
    'total_traffic', 'organic_traffic', 'ad_traffic',
    'ad_impressions', 'ad_spends'
]

for col in cols_to_clean:
    df[col] = df[col].astype(str).str.replace(',', '').astype(float)

#Check the result
print(df.dtypes)
print(df.head())
```

```
product_code              object
date              datetime64[ns]
total_sales              float64
organic_sales            float64
ad_sales                 float64
total_traffic            float64
organic_traffic          float64
ad_traffic               float64
overall_units              int64
organic_units            float64
ad_units                 float64
ad_impressions           float64
ad_spends                float64
dtype: object
  product_code       date  total_sales  organic_sales  ad_sales  \
0  B07F5NCTN28 2024-04-01      31500.0        13848.0   11262.0
1  B07R3ZKB7D8 2024-04-01      26680.0        16399.0    6436.0
2  B07F5M62172 2024-04-01      26600.0         9158.0   13452.0
3  B07P8FP14D3 2024-04-01      12780.0         6919.0    5330.0
4  B07F5LZHVL1 2024-04-01      11960.0        10764.0       0.0

   total_traffic  organic_traffic  ad_traffic  overall_units  organic_units  \
0         1548.0          1279.55      268.45             90           52.0
1          774.0           677.58       96.42             58           35.0
2         1086.0           822.23      263.77             95           39.0
3          626.0           492.41      133.59             71           38.0
4          154.0           153.18        0.82             52           52.0

   ad_units  ad_impressions  ad_spends
0      38.0         72433.0     5561.0
1      23.0         24171.0     2186.0
2      56.0         41932.0     4935.0
3      33.0         26649.0     1089.0
4       0.0           123.0        5.0
```

In [4]:
```
##step2
#Filter April and May data
april = df[df['date'].dt.month == 4]
may = df[df['date'].dt.month == 5]

#Aggregate by month
def compute_metrics(data):
    total_sales = data['total_sales'].sum()
    overall_units = data['overall_units'].sum()
    total_traffic = data['total_traffic'].sum()
    ad_spends = data['ad_spends'].sum()
    ad_sales = data['ad_sales'].sum()

    conversion_rate = overall_units / total_traffic if total_traffic else 0
    asp = total_sales / overall_units if overall_units else 0
    roas = ad_sales / ad_spends if ad_spends else 0
    acos = ad_spends / ad_sales if ad_sales else 0

    return {
        'Revenue (₹)': total_sales,
        'Units Sold': overall_units,
```

```
        'Traffic': total_traffic,
        'Conversion Rate': conversion_rate,
        'ASP (₹)': asp,
        'Ad Spend (₹)': ad_spends,
        'ROAS': roas,
        'ACOS': acos
    }

april_metrics = compute_metrics(april)
may_metrics = compute_metrics(may)

# alculate deltas and % change
mom_table = pd.DataFrame([april_metrics, may_metrics], index=['April', 'May'])
mom_table.loc['^ (Change)'] = mom_table.loc['May'] - mom_table.loc['April']
mom_table.loc['% Change'] = ((mom_table.loc['May'] - mom_table.loc['April']) / mom_table.loc['April']) * 100

# Show result
print(mom_table)
```

```
             Revenue (₹)    Units Sold       Traffic  Conversion Rate  \
April       9.271914e+06  30755.000000  262250.00000         0.117274
May         9.354981e+06  30108.000000  222669.00000         0.135214
^ (Change)  8.306708e+04   -647.000000  -39581.00000         0.017941
% Change    8.959001e-01     -2.103723     -15.09285        15.298037

               ASP (₹)   Ad Spend (₹)       ROAS       ACOS
April       301.476623  832984.000000   2.742891   0.364579
May         310.714117  723522.000000   3.287223   0.304208
^ (Change)    9.237494 -109462.000000   0.544332  -0.060371
% Change      3.064083     -13.140949  19.845201 -16.559029
```

In [5]:

```
##step3
import numpy as np
# Add helper function to calculate decomposition for any stream
def decompose_sales_change(traffic, units, sales):
    CR_apr = april[units].sum() / april[traffic].sum()
    CR_may = may[units].sum() / may[traffic].sum()
    ASP_apr = april[sales].sum() / april[units].sum()
    ASP_may = may[sales].sum() / may[units].sum()
    T_apr = april[traffic].sum()
    T_may = may[traffic].sum()

    # Chain rule log-decomposition
    Δ_log_sales = np.log(T_may * CR_may * ASP_may) - np.log(T_apr * CR_apr * ASP_apr)
    Δ_log_traffic = np.log(T_may) - np.log(T_apr)
    Δ_log_cr = np.log(CR_may) - np.log(CR_apr)
    Δ_log_asp = np.log(ASP_may) - np.log(ASP_apr)

    return {
        'Traffic %': (Δ_log_traffic / Δ_log_sales) * 100,
        'Conversion Rate %': (Δ_log_cr / Δ_log_sales) * 100,
        'ASP %': (Δ_log_asp / Δ_log_sales) * 100
    }

#Run for Organic Sales
organic_decomp = decompose_sales_change('organic_traffic', 'organic_units', 'organic_sales')

#Run for Ad Sales
ad_decomp = decompose_sales_change('ad_traffic', 'ad_units', 'ad_sales')

#Display both
print(" Organic Sales Decomposition:")
print(pd.DataFrame([organic_decomp]))

print("\n Ad Sales Decomposition:")
print(pd.DataFrame([ad_decomp]))
```

```
 Organic Sales Decomposition:
     Traffic %  Conversion Rate %       ASP %
0  1193.425867        -921.735531 -171.690337

 Ad Sales Decomposition:
    Traffic %  Conversion Rate %      ASP %
0 -458.283742         490.427938  67.855804
```

In [7]:

```
##step4
#Add ASP and Conversion columns
df['conversion_rate'] = df['overall_units'] / df['total_traffic']
df['asp'] = df['total_sales'] / df['overall_units']

#Separate April & May again
april_df = df[df['date'].dt.month == 4].copy()
may_df = df[df['date'].dt.month == 5].copy()
```

```python
#Group by ASIN (Product Code) and calculate aggregates
def asin_monthly_summary(data):
    return data.groupby('product_code').agg({
        'total_sales': 'sum',
        'total_traffic': 'sum',
        'overall_units': 'sum',
        'conversion_rate': 'mean',
        'asp': 'mean'
    }).rename(columns={
        'total_sales': 'sales',
        'total_traffic': 'traffic',
        'overall_units': 'units'
    })

april_summary = asin_monthly_summary(april_df)
may_summary = asin_monthly_summary(may_df)

#Calculate deltas
asin_change = may_summary - april_summary
asin_change['conversion_rate_change'] = may_summary['conversion_rate'] - april_summary['conversion_rate']
asin_change['asp_change'] = may_summary['asp'] - april_summary['asp']

#Top ASINs by drop in Conversion Rate
top_cr_drop = asin_change.sort_values('conversion_rate_change').head(10)

#Top ASINs by drop in Traffic
top_traffic_drop = asin_change.sort_values('traffic').head(10)
#Outliers: ASINs with traffic > 1000 and units == 0
outliers = may_df.groupby('product_code').agg({
    'total_traffic': 'sum',
    'overall_units': 'sum'
}).query('total_traffic > 1000 and overall_units == 0')

print(" Top 10 ASINs with Conversion Rate Drop:\n", top_cr_drop[['conversion_rate_change']])
print("\n Top 10 ASINs with Traffic Drop:\n", top_traffic_drop[['traffic']])
print("\n Outlier ASINs (High Traffic, Zero Units):\n", outliers)
```

```
 Top 10 ASINs with Conversion Rate Drop:
              conversion_rate_change
product_code
B07R3JJGYW2                    -inf
B009YDXV9Q4                    -inf
B07MSPKHCC7                    -inf
B07R4TV4SF5                    -inf
B07P6CCH3R5                    -inf
B07R4TK5RB7                    -inf
B07P9K24MW1                    -inf
B07P8FP3MN7                    -inf
B07HZQN4SF7                    -inf
B0823RGLLS2                    -inf

 Top 10 ASINs with Traffic Drop:
              traffic
product_code
B07F5NCTN28  -12169.0
B07R3ZKB7D8   -4050.0
B0DNMQVKFL5   -3677.0
B07P8FP14D3   -2647.0
B07F5M62172   -2117.0
B07R4WTRN21   -1890.0
B07CGNCPKT6   -1780.0
B07PCPQ57N6   -1691.0
B07PBJNS5X9   -1523.0
B07MQ63G912   -1512.0

 Outlier ASINs (High Traffic, Zero Units):
 Empty DataFrame
Columns: [total_traffic, overall_units]
Index: []
```

In [ ]: