E-commerce MoM Analysis

Period: April 2024 vs May 2024 Objectives:

1. Clean dataset with derived metrics
2. MoM KPIs with % change and absolute delta
3. Magic Equation Decomposition
4. Root Cause Analysis (Five Whys)
5. MECE Hypotheses and Product Drivers

In [227…
```
import pandas as pd
import numpy as np

df = pd.read_csv('/Users/ankit/Downloads/Amazon Sales data - Master data.csv


print(df.shape)
print(df.info())

# convert date column
df['Date']=pd.to_datetime(df['Date'])
```

```
(8631, 13)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8631 entries, 0 to 8630
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Product Code     8631 non-null   object
 1   Date             8631 non-null   object
 2   Total_Sales      8631 non-null   object
 3   Organic_Sales    8631 non-null   object
 4   Ad_Sales         8631 non-null   object
 5   Total_Traffic    8631 non-null   object
 6   Organic Traffic  8631 non-null   object
 7   Ad traffic       8631 non-null   object
 8   Overall_Units    8631 non-null   int64
 9   Organic_Units    8631 non-null   float64
 10  Ad_Units         8631 non-null   float64
 11  Ad_Impressions   8631 non-null   object
 12  Ad_Spends        8631 non-null   object
dtypes: float64(2), int64(1), object(10)
memory usage: 876.7+ KB
None
```

In [228…
```
# Renaming columns for uniformity
df.rename(columns={'Ad traffic': 'Ad_Traffic'}, inplace=True)
df.rename(columns={'Organic Traffic': 'Organic_Traffic'}, inplace=True)
```

In [229…
```
df.head()
```

Out[229]:

	Product Code	Date	Total_Sales	Organic_Sales	Ad_Sales	Total_Traffic	Organic_Traff
0	B07F5NCTN28	2024-04-01	31,500	13,848	11,262	1,548	1,279.5
1	B07R3ZKB7D8	2024-04-01	26,680	16,399	6,436	774	677.5
2	B07F5M62172	2024-04-01	26,600	9,158	13,452	1,086	822.2
3	B07P8FP14D3	2024-04-01	12,780	6,919	5,330	626	492.4
4	B07F5LZHVL1	2024-04-01	11,960	10,764	0	154	153.1

In [230…

```
# List all numeric columns that have commas
columns_with_commas = [
    'Total_Sales',
    'Organic_Sales',
    'Ad_Sales',
    'Total_Traffic',
    'Organic_Traffic',
    'Ad_Traffic',
    'Ad_Impressions',
    'Ad_Spends'
]

# Remove commas & convert to float
for col in columns_with_commas:
    df[col] = df[col].astype(str).str.replace(',', '')
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

In [231…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8631 entries, 0 to 8630
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Product Code     8631 non-null   object
 1   Date             8631 non-null   datetime64[ns]
 2   Total_Sales      8631 non-null   float64
 3   Organic_Sales    8631 non-null   int64
 4   Ad_Sales         8631 non-null   int64
 5   Total_Traffic    8631 non-null   int64
 6   Organic_Traffic  8631 non-null   float64
 7   Ad_Traffic       8631 non-null   float64
 8   Overall_Units    8631 non-null   int64
 9   Organic_Units    8631 non-null   float64
 10  Ad_Units         8631 non-null   float64
 11  Ad_Impressions   8631 non-null   int64
 12  Ad_Spends        8631 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(6), object(1)
memory usage: 876.7+ KB
```

In [232…

```
# Filling missing values with 0 for spends, units, traffic and impressions
df['Ad_Spends'] = df['Ad_Spends'].fillna(0)
df['Ad_Impressions'] = df['Ad_Impressions'].fillna(0)
df['Ad_Traffic'] = df['Ad_Traffic'].fillna(0)
df['Organic_Traffic'] = df['Organic_Traffic'].fillna(0)
df['Ad_Sales'] = df['Ad_Sales'].fillna(0)
df['Organic_Sales'] = df['Organic_Sales'].fillna(0)
```

```
df['Ad_Units'] = df['Ad_Units'].fillna(0)
df['Organic_Units'] = df['Organic_Units'].fillna(0)
```

In [233…

```
# Fixing negative Values
cols_non_negative = [
    'Total_Sales', 'Organic_Sales', 'Ad_Sales',
    'Total_Traffic', 'Organic_Traffic', 'Ad_Traffic',
    'Overall_Units', 'Organic_Units', 'Ad_Units',
    'Ad_Impressions'
]

print("Negative Values Check:")
for col in cols_non_negative:
    neg_count = (df[col] < 0).sum()
    if neg_count > 0:
        print(f"  - {col}: {neg_count} negatives found")

#Clip negatives to zero
for col in cols_non_negative:
    df[col] = df[col].clip(lower=0)

print("Negative values clipped to zero")
```

```
Negative Values Check:
  - Total_Sales: 118 negatives found
  - Organic_Sales: 345 negatives found
  - Organic_Traffic: 4 negatives found
  - Overall_Units: 118 negatives found
  - Organic_Units: 371 negatives found
Negative values clipped to zero
```

In [234…

```
from scipy import stats
import numpy as np

# Select numeric columns only
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()

outliers = {}

print("Outlier Check (Z-Score > 3 or < -3):")
for col in num_cols:
    z_scores = stats.zscore(df[col])
    outlier_mask = np.abs(z_scores) > 3
    outlier_count = outlier_mask.sum()
    if outlier_count > 0:
        outliers[col] = outlier_count
        print(f"  - {col}: {outlier_count} potential outliers")
```

```
Outlier Check (Z-Score > 3 or < -3):
  - Total_Sales: 189 potential outliers
  - Organic_Sales: 189 potential outliers
  - Ad_Sales: 239 potential outliers
  - Total_Traffic: 192 potential outliers
  - Organic_Traffic: 196 potential outliers
  - Ad_Traffic: 125 potential outliers
  - Overall_Units: 212 potential outliers
  - Organic_Units: 180 potential outliers
  - Ad_Units: 261 potential outliers
  - Ad_Impressions: 82 potential outliers
  - Ad_Spends: 179 potential outliers
```

In [235…

```
# Calculating derived metrics
df['ROAS']=np.where(df['Ad_Spends'] > 0, df['Ad_Sales'] / df['Ad_Spends'], r
df['ACoS'] = np.where(df['Ad_Sales'] > 0, df['Ad_Spends'] / df['Ad_Sales'],
```

```
df['CTR'] = np.where(df['Ad_Impressions'] > 0, df['Ad_Traffic'] / df['Ad_Imp

df['Conv_Rate'] = np.where(df['Total_Traffic'] > 0, df['Overall_Units'] / d
df['ASP'] = np.where(df['Overall_Units'] > 0, df['Total_Sales'] / df['Overa

df['Variance_Sales'] = df['Total_Sales'] - (df['Organic_Sales'] + df['Ad_Sa
df['Variance_Traffic'] = df['Total_Traffic'] - (df['Organic_Traffic'] + df[
df['Variance_Units'] = df['Overall_Units'] - (df['Organic_Units'] + df['Ad_U

df.head()
```

Out[235]:

	Product Code	Date	Total_Sales	Organic_Sales	Ad_Sales	Total_Traffic	Organic_Traff
0	B07F5NCTN28	2024-04-01	31500.0	13848	11262	1548	1279.5
1	B07R3ZKB7D8	2024-04-01	26680.0	16399	6436	774	677.5
2	B07F5M62172	2024-04-01	26600.0	9158	13452	1086	822.2
3	B07P8FP14D3	2024-04-01	12780.0	6919	5330	626	492.4
4	B07F5LZHVL1	2024-04-01	11960.0	10764	0	154	153.1

5 rows × 21 columns

In [236…

```
# Exporting clean dataset
df.to_csv('cleaned_amazon_sales_dataset.csv', index=False)
```

In [237…

```
# Setting up date ranges for MoM calculations
apr_start = '2024-04-01'
apr_end = '2024-04-30'
may_start = '2024-05-01'
may_end = '2024-05-31'

df_apr = df[(df['Date'] >= apr_start) & (df['Date'] <= apr_end)]
df_may = df[(df['Date'] >= may_start) & (df['Date'] <= may_end)]

print(df_apr.shape, df_may.shape)
```

```
(4242, 21) (4389, 21)
```

In [238…

```
# Defining aggregator
def calc_kpis(df_subset):
    total_sales = df_subset['Total_Sales'].sum()
    units = df_subset['Overall_Units'].sum()
    traffic = df_subset['Total_Traffic'].sum()
    conv_rate = units / traffic if traffic != 0 else np.nan
    asp = total_sales / units if units != 0 else np.nan
    ad_spend = df_subset['Ad_Spends'].sum()
    ad_sales = df_subset['Ad_Sales'].sum()
    roas = ad_sales / ad_spend if ad_spend != 0 else np.nan
    acos = ad_spend / ad_sales if ad_sales != 0 else np.nan

    organic_sales = df_subset['Organic_Sales'].sum()
    organic_units = df_subset['Organic_Units'].sum()
    organic_traffic = df_subset['Organic_Traffic'].sum()

    ad_units = df_subset['Ad_Units'].sum()
    ad_traffic = df_subset['Ad_Traffic'].sum()
```

```
    return {
        'Revenue': total_sales,
        'Units': units,
        'Traffic': traffic,
        'Conversion Rate': conv_rate,
        'ASP': asp,
        'Ad Spend': ad_spend,
        'ROAS': roas,
        'ACoS': acos,
        'Organic_Sales': organic_sales,
        'Organic_Units': organic_units,
        'Organic_Traffic': organic_traffic,
        'Ad_Sales': ad_sales,
        'Ad_Units': ad_units,
        'Ad_Traffic': ad_traffic
    }

# Getting aggregated KPIs for both months
kpi_apr = calc_kpis(df_apr)
kpi_may = calc_kpis(df_may)

# MoM Table
kpi_df = pd.DataFrame({'April': kpi_apr, 'May': kpi_may}).T
kpi_df = kpi_df.reset_index().rename(columns={'index': 'Month'})

# Add %change and absolute change
kpi_df = kpi_df.set_index('Month').T
kpi_df['Absolute Change'] = kpi_df['May'] - kpi_df['April']
kpi_df['% MoM Change'] = np.where(kpi_df['April'] != 0, (kpi_df['May'] - kp

print(kpi_df)
```

Month	April	May	Absolute Change	% MoM Change
Revenue	9.298581e+06	9.395085e+06	96503.320000	1.037828
Units	3.085100e+04	3.024500e+04	-606.000000	-1.964280
Traffic	2.622500e+05	2.226690e+05	-39581.000000	-15.092850
Conversion Rate	1.176397e-01	1.358294e-01	0.018190	15.462267
ASP	3.014029e+02	3.106327e+02	9.229740	3.062260
Ad Spend	8.329840e+05	7.235220e+05	-109462.000000	-13.140949
ROAS	2.742891e+00	3.287223e+00	0.544332	19.845201
ACoS	3.645789e-01	3.042082e-01	-0.060371	-16.559029
Organic_Sales	5.623981e+06	5.589609e+06	-34372.000000	-0.611168
Organic_Units	2.245668e+04	2.176378e+04	-692.900000	-3.085496
Organic_Traffic	2.102226e+05	1.793904e+05	-30832.180000	-14.666447
Ad_Sales	2.284784e+06	2.378378e+06	93594.000000	4.096405
Ad_Units	9.159619e+03	9.278590e+03	118.971000	1.298864
Ad_Traffic	5.202843e+04	4.328482e+04	-8743.612000	-16.805449

Revenue went up slightly by 1%, However the units sold declined by 2% and Traffic also dropped by 15%.

The Revenue did not tanked due to increase in traffic conversion rate by 15% and slight jump in Average selling price by 3%

Ad expenditure dropped by 13% but return on Ads jumped by 20% and Cost of Ads dropped by 16.5%

We sold fewer units to fewer visitors but higher prices and better conversion helped to offset the drop. Advertising was efficient with less expenditure and more returns

Magic Equation

Sales = Traffic X Conv Rate X ASP

In [239…

```
T0 = kpi_apr['Traffic']
C0 = kpi_apr['Conversion Rate']
A0 = kpi_apr['ASP']

T1 = kpi_may['Traffic']
C1 = kpi_may['Conversion Rate']
A1 = kpi_may['ASP']

#Traffic effect
traffic_effect = (T1 - T0) * C0 * A0

#Conversion effect
conv_effect = T1 * (C1 - C0) * A0

#ASP effect
asp_effect = T1 * C1 * (A1 - A0)

total_delta = traffic_effect + conv_effect + asp_effect

print(f'Traffic Effect: {traffic_effect:.2f}')
print(f'Conversion Effect: {conv_effect:.2f}')
print(f'ASP Effect: {asp_effect:.2f}')
print(f'Total Explained Delta: {total_delta:.2f}')

#Cross-check
actual_delta = kpi_may['Revenue'] - kpi_apr['Revenue']
print(f'Actual Sales Delta: {actual_delta:.2f}')
```

```
Traffic Effect: -1403420.98
Conversion Effect: 1220770.81
ASP Effect: 279153.49
Total Explained Delta: 96503.32
Actual Sales Delta: 96503.32
```

Our sales remained flat for the month of May after the drop in traffic due to jump in the conversion and mild increase of the Average selling price

Decomposition for Organic Sales

In [240…

```
#Drivers

T2 = kpi_apr['Organic_Traffic']
U2 = kpi_apr['Organic_Units']
S2 = kpi_apr['Organic_Sales']
#conversion rate
C2 = U2 / T2
#Average selling price
A2 = S2 / U2

T3 = kpi_may['Organic_Traffic']
U3 = kpi_may['Organic_Units']
S3 = kpi_may['Organic_Sales']
#conversion rate
```

```
C3 = U3 / T3
#Average selling price
A3 = S3 / U3

#Traffic effect
traffic_effect = (T3 - T2) * C2 * A2

#Conversion effect
conversion_effect = T3 * (C3 - C2) * A2

#ASP effect
asp_effect = T3 * C3 * (A3 - A2)

explained_delta = traffic_effect + conversion_effect + asp_effect
actual_delta = S3 - S2

print("Organic Decomposition:")
print(f"Traffic Effect: {traffic_effect:.2f}")
print(f"Conversion Effect: {conversion_effect:.2f}")
print(f"ASP Effect: {asp_effect:.2f}")
print(f"Explained Delta: {explained_delta:.2f}")
print(f"Actual Delta: {actual_delta:.2f}")
```

```
Organic Decomposition:
Traffic Effect: -824838.18
Conversion Effect: 651310.47
ASP Effect: 139155.72
Explained Delta: -34372.00
Actual Delta: -34372.00
```

Organic Traffic dropped but conversion increased supported by increase in average selling price resulted into drop in organic sales

organic traffic dragged sales

Decomposition of Ad Sales

In [241…

```
#Drivers

T4 = kpi_apr['Ad_Traffic']
U4 = kpi_apr['Ad_Units']
S4 = kpi_apr['Ad_Sales']
#conversion rate
C4 = U4 / T4
#Average selling price
A4 = S4 / U4

T5 = kpi_may['Ad_Traffic']
U5 = kpi_may['Ad_Units']
S5 = kpi_may['Ad_Sales']
#conversion rate
C5 = U5 / T5
#Average selling price
A5 = S5 / U5

# Decompose
traffic_effect = (T5 - T4) * C4 * A4
conversion_effect = T5 * (C5 - C4) * A4
asp_effect = T5 * C5 * (A5 - A4)

explained_delta = traffic_effect + conversion_effect + asp_effect
```

```
actual_delta = S5 - S4

print("Ad Sales Decomposition:")
print(f"Traffic Effect: {traffic_effect:.2f}")
print(f"Conversion Effect: {conversion_effect:.2f}")
print(f"ASP Effect: {asp_effect:.2f}")
print(f"Explained Delta: {explained_delta:.2f}")
print(f"Actual Delta: {actual_delta:.2f}")
```

```
Ad Sales Decomposition:
Traffic Effect: -383968.21
Conversion Effect: 413644.44
ASP Effect: 63917.76
Explained Delta: 93594.00
Actual Delta: 93594.00
```

Ad traffic dropped a lot but conversion increased which was supported by better average selling price resulting into increase in Ad sales

Ads became efficient (Higher conversion, Better ASP)

In [242…

```
# Example: Top 10 Traffic delta drivers
traffic_by_asin_apr = df_apr.groupby('Product Code')['Total_Traffic'].sum()
traffic_by_asin_may = df_may.groupby('Product Code')['Total_Traffic'].sum()
traffic_delta = (traffic_by_asin_may - traffic_by_asin_apr).sort_values(asce
print(traffic_delta.head(10))
```

```
Product Code
B07P6CBN1W6    2244.0
B07R3ZKHYQ9     837.0
B0855D88PV4     810.0
B07F5RZWH46     766.0
B07F5M3K1Y9     522.0
B07PBLHPN85     429.0
B07R3ZHZDL4     407.0
B07HKF39CM8     363.0
B07R4XJ3HY1     305.0
B07MJBYFQY9     213.0
Name: Total_Traffic, dtype: float64
```

Root Casuse Analysis

Drivers

1. Organic traffic dropped by 15%
2. Ad traffic down
3. Conversion improvedb

Organic traffic drop is the negative driver

In [243…

```
#Calculating organic Traffic by ASIN

organic_traffic_apr = df_apr.groupby('Product Code')['Organic_Traffic'].sum(
organic_traffic_may = df_may.groupby('Product Code')['Organic_Traffic'].sum(

#Merging for MoM
organic_traffic_mom = organic_traffic_apr.merge(
    organic_traffic_may,
    on='Product Code',
    suffixes=('_Apr', '_May')
```

```
)

organic_traffic_mom['Abs_Change'] = organic_traffic_mom['Organic_Traffic_May
organic_traffic_mom['Pct_Change'] = organic_traffic_mom['Abs_Change']*100 /

#Sorting for biggest drops
organic_traffic_mom = organic_traffic_mom.sort_values('Pct_Change')

organic_traffic_mom.head(10)
```

Out[243]:

	Product Code	Organic_Traffic_Apr	Organic_Traffic_May	Abs_Change	Pct_Change
35	B07MJBYM6K2	4.00	0.00	-4.00	-100.000000
134	B07VQQ4QTM2	75.03	0.00	-75.03	-100.000000
110	B07R4V5W3K9	320.00	55.00	-265.00	-82.812500
117	B07R4W54L84	1712.00	333.00	-1379.00	-80.549065
47	B07MSPG7NJ4	241.00	70.00	-171.00	-70.954357
125	B07R4WTRN21	2906.69	930.94	-1975.75	-67.972505
111	B07R4V79GY6	70.00	23.00	-47.00	-67.142857
56	B07P8FLW976	12.00	4.00	-8.00	-66.666667
119	B07R4WBMSW6	158.66	58.34	-100.32	-63.229547
19	B07HZCYKFM1	591.07	249.00	-342.07	-57.873010

In [244…

```
#Calculating Conversion for April
apr_cr = df_apr.groupby('Product Code').apply(
    lambda x: x['Organic_Units'].sum() / x['Organic_Traffic'].sum() if x['Or
).reset_index(name='Conv_Apr')

#Calculating Conversion for May
may_cr = df_may.groupby('Product Code').apply(
    lambda x: x['Organic_Units'].sum() / x['Organic_Traffic'].sum() if x['Or
).reset_index(name='Conv_May')

#Merge
organic_traffic_mom = organic_traffic_mom.merge(apr_cr, on='Product Code', h
organic_traffic_mom = organic_traffic_mom.merge(may_cr, on='Product Code', h

#Calculate CR change
organic_traffic_mom['Conv_Change'] = organic_traffic_mom['Conv_May'] - organ


organic_traffic_mom.sort_values('Pct_Change').head(10)
```

Out[244]:

	Product Code	Organic_Traffic_Apr	Organic_Traffic_May	Abs_Change	Pct_Change	C
0	B07MJBYM6K2	4.00	0.00	-4.00	-100.000000	0
1	B07VQQ4QTM2	75.03	0.00	-75.03	-100.000000	0
2	B07R4V5W3K9	320.00	55.00	-265.00	-82.812500	0
3	B07R4W54L84	1712.00	333.00	-1379.00	-80.549065	0
4	B07MSPG7NJ4	241.00	70.00	-171.00	-70.954357	0
5	B07R4WTRN21	2906.69	930.94	-1975.75	-67.972505	0
6	B07R4V79GY6	70.00	23.00	-47.00	-67.142857	0
7	B07P8FLW976	12.00	4.00	-8.00	-66.666667	0
8	B07R4WBMSW6	158.66	58.34	-100.32	-63.229547	0
9	B07HZCYKFM1	591.07	249.00	-342.07	-57.873010	0

In [245…

```
#Merging Organic Units
units_mom = organic_units_apr.merge(
    organic_units_may,
    on='Product Code',
    suffixes=('_Apr', '_May')
)
organic_traffic_mom = organic_traffic_mom.merge(units_mom, on='Product Code

#Merging Ad Units
ad_units_mom = ad_units_apr.merge(
    ad_units_may,
    on='Product Code',
    suffixes=('_Apr', '_May')
)
organic_traffic_mom = organic_traffic_mom.merge(ad_units_mom, on='Product Co

#Calculating change
organic_traffic_mom['Organic_Units_Change'] = organic_traffic_mom['Organic_U
organic_traffic_mom['Ad_Units_Change'] = organic_traffic_mom['Ad_Units_May']


organic_traffic_mom.head(10)
```

Out[245]:

	Product Code	Organic_Traffic_Apr	Organic_Traffic_May	Abs_Change	Pct_Change	C
0	B07MJBYM6K2	4.00	0.00	-4.00	-100.000000	0
1	B07VQQ4QTM2	75.03	0.00	-75.03	-100.000000	0
2	B07R4V5W3K9	320.00	55.00	-265.00	-82.812500	0
3	B07R4W54L84	1712.00	333.00	-1379.00	-80.549065	0
4	B07MSPG7NJ4	241.00	70.00	-171.00	-70.954357	0
5	B07R4WTRN21	2906.69	930.94	-1975.75	-67.972505	0
6	B07R4V79GY6	70.00	23.00	-47.00	-67.142857	0
7	B07P8FLW976	12.00	4.00	-8.00	-66.666667	0
8	B07R4WBMSW6	158.66	58.34	-100.32	-63.229547	0
9	B07HZCYKFM1	591.07	249.00	-342.07	-57.873010	0

1. A chunk of traffic shifted to ASINs with lower average conversion
2. The higher-conversion ASINs lost sessions

Some ASINs (B07VQQ4QTM2, B07R4V5W3K9) lost BOTH traffic and conversion

Organic Conversion dropped by 15%: Because traffic mix shifted away from high-conversion ASINs.

1. Some high-performing ASINs (like B07R4W54L84) saw traffic collapse
2. Other ASINs with poor conversion didn't pick up enough

In [246…

```
#Listing top ASINs with largest organic traffic drop
top_traffic_loss = organic_traffic_mom.sort_values('Pct_Change').head(10)['P

#Subset original data frame for above ASINs only
df_top_loss_apr = df_apr[df_apr['Product Code'].isin(top_traffic_loss)]
df_top_loss_may = df_may[df_may['Product Code'].isin(top_traffic_loss)]

#Grouping stats for comparison
traffic_check_apr = df_top_loss_apr.groupby('Product Code').agg({
    'Total_Traffic': 'sum',
    'Ad_Traffic': 'sum',
    'Organic_Traffic': 'sum',
    'Overall_Units': 'sum'
}).reset_index().rename(columns={
    'Total_Traffic': 'Total_Traffic_Apr',
    'Ad_Traffic': 'Ad_Traffic_Apr',
    'Organic_Traffic': 'Organic_Traffic_Apr',
    'Overall_Units': 'Units_Apr'
})

traffic_check_may = df_top_loss_may.groupby('Product Code').agg({
    'Total_Traffic': 'sum',
    'Ad_Traffic': 'sum',
    'Organic_Traffic': 'sum',
    'Overall_Units': 'sum'
}).reset_index().rename(columns={
    'Total_Traffic': 'Total_Traffic_May',
    'Ad_Traffic': 'Ad_Traffic_May',
    'Organic_Traffic': 'Organic_Traffic_May',
    'Overall_Units': 'Units_May'
})

#Merging it for both months
traffic_check = traffic_check_apr.merge(
    traffic_check_may, on='Product Code', how='outer'
)

#Calculating deltas
traffic_check['Total_Traffic_Change'] = traffic_check['Total_Traffic_May'] -
traffic_check['Ad_Traffic_Change'] = traffic_check['Ad_Traffic_May'] - traf
traffic_check['Units_Change'] = traffic_check['Units_May'] - traffic_check[

#Clean output
traffic_check = traffic_check.sort_values('Total_Traffic_Change')

traffic_check.head(10)
```

Out[246]:

	Product Code	Total_Traffic_Apr	Ad_Traffic_Apr	Organic_Traffic_Apr	Units_Apr	Total
8	B07R4WTRN21	3073	166.31	2906.69	235	
6	B07R4W54L84	1712	0.00	1712.00	436	
0	B07HZCYKFM1	610	18.93	591.07	25	
4	B07R4V5W3K9	320	0.00	320.00	130	
2	B07MSPG7NJ4	241	0.00	241.00	11	
7	B07R4WBMSW6	162	3.35	158.66	12	
9	B07VQQ4QTM2	76	0.97	75.03	11	
5	B07R4V79GY6	70	0.00	70.00	19	
3	B07P8FLW976	12	0.00	12.00	0	
1	B07MJBYM6K2	4	0.00	4.00	1	

1. B07R4WTRN21 show Total_Traffic down & Ad_Traffic up slightly. Organic down big but Ad_Traffic up = possible budget shift
2. B07R4W54L84 shows Ad Traffic both flat, pure organic loss. No Ad_Traffic in either month, so drop is likely rank or availability
3. B07R4W54L84 Units dropped heavily high chance it ran out of stock or got de-prioritized.
4. B07MJBYM6K2 fully went to zero in may, listing paused or stockout

In [247…

```
#Flagging ASINs with big traffic drop and units dropped to zero or near to z
rca_flags = traffic_check[
    (traffic_check['Total_Traffic_Change'] < -50) &
    (traffic_check['Units_May'] <= 10)
]

rca_flags[['Product Code', 'Total_Traffic_Apr', 'Total_Traffic_May',
                  'Units_Apr', 'Units_May', 'Ad_Traffic_Apr', 'Ad_Traffic_May
```

Out[247]:

	Product Code	Total_Traffic_Apr	Total_Traffic_May	Units_Apr	Units_May	Ad_Traffic_
2	B07MSPG7NJ4	241	70	11	9	0
7	B07R4WBMSW6	162	64	12	5	3
9	B07VQQ4QTM2	76	0	11	0	0

1. B07MSPG7NJ4: Traffic dropped by 71% Ad traffic stayed 0. Possible RCA: Loss of organic exposure. Maybe listing rank dropped or OOS periods.
2. B07R4WBMSW6: Traffic dropped by 60% units dropped more sharply (12 to 5) Possible RCA: Paid ads did not offset the organic drop, check keyword bids & relevancy.
3. B07VQQ4QTM2: Organic traffic and units lost, Ad traffic went from 0.97 to 3.00. Possible RCA: Possible OOS (stockout), delisted, or suspended ASIN. Full zero units

strongly hints stockout.

MECE Hypothesis Framing for Organic Conversion Drop (15%)

Hypothesis	Expected Supporting Signals	Additional Data Needed	Quick Test Plan
Out-of-Stock	Days with zero units sold despite traffic	Daily inventory status by ASIN	Plot units vs stock; flag days with traffic but zero stock
Listing Issues	Sudden drop in traffic or conversion, no clear stock issue	Listing status logs (active, suppressed, stranded)	Cross-check with seller account status reports
Negative Review Spike	Increase in 1-star reviews, drop in average rating	MoM review data: counts, ratings, sentiment	Trend avg. rating vs conversion by ASIN
Increased ASP vs Competitors	ASP up while competitor prices flat or down	Competitor pricing data	Compare ASP trend vs competitors, plot against conversion
Low Relevant Traffic	Higher traffic but lower conversion; irrelevant clicks	keyword data	Look at keyword performance shift MoM
Competitor Promotion	Drop in share of impressions	Market share, impression share, ad auction data	Compare ad share and keyword auction results
Seasonality or External Factor	Same pattern last year / industry trend	Historical data (YoY), market reports	Compare YoY trend, check same period last year

Product Level Diagnostics

In [248…

```
#Computing the contribution of each ASIN's conversion rate change
organic_traffic_mom['Conv_Impact'] = organic_traffic_mom['Conv_Change'] * or

#Ranking in descending order by negative impact
top_conv_drop_asins = organic_traffic_mom.sort_values(by='Conv_Impact').hea

#Result
top_conv_drop_asins[['Product Code', 'Conv_Change','Conv_Impact', 'Organic_
```

Out[248]:

	Product Code	Conv_Change	Conv_Impact	Organic_Traffic_Apr	Organic_Traffic_May
94	B07R3ZKHYQ9	-0.071279	-230.065569	3227.68	3196.88
112	B07R3TV8M37	-0.086526	-212.818027	2459.58	2707.00
110	B07P6CBN1W6	-0.051501	-190.527166	3699.49	3974.27
84	B07F5LZHVL1	-0.035350	-145.658902	4120.48	3904.35
36	B07R4VDDG12	-0.070950	-104.165098	1468.15	1084.59
102	B0855D88PV4	-0.060596	-91.501410	1510.02	1552.39
115	B07F5M3K1Y9	-0.058574	-85.810258	1464.99	1625.02
68	B07MSNKFY56	-0.029277	-69.982235	2390.31	2111.75
2	B07R4V5W3K9	-0.151705	-48.545455	320.00	55.00
113	B0823RGLLS2	-0.033082	-47.442340	1434.07	1582.99

It shows top ASINs contributing to conversion loss

In [249…

```
#Calculating ASP for April and May
organic_traffic_mom = organic_traffic_mom.merge(df_apr.groupby('Product Code
organic_traffic_mom = organic_traffic_mom.merge(df_may.groupby('Product Code

#Calculating ASP Change
organic_traffic_mom['ASP_Change'] = organic_traffic_mom['ASP_May'] - organic

#Flags
organic_traffic_mom['Large_Traffic_Drop'] = organic_traffic_mom['Pct_Change
organic_traffic_mom['Large_ASP_Change'] = abs(organic_traffic_mom['ASP_Chang
organic_traffic_mom['Possible_Stockout'] = (organic_traffic_mom['Organic_Tra

#Combining the flagged result
organic_traffic_mom['Is_Outlier'] = organic_traffic_mom[['Large_Traffic_Drop

#Filtering outliers
outliers = organic_traffic_mom[organic_traffic_mom['Is_Outlier']]

outliers[['Product Code', 'Pct_Change', 'ASP_Change', 'Organic_Traffic_May',
```

Out[249]:

	Product Code	Pct_Change	ASP_Change	Organic_Traffic_May	Organic_Units_May	I
0	B07MJBYM6K2	-100.000000	NaN	0.00	0.00	
1	B07VQQ4QTM2	-100.000000	NaN	0.00	0.00	
2	B07R4V5W3K9	-82.812500	0.0	55.00	14.00	
3	B07R4W54L84	-80.549065	0.0	333.00	184.00	
4	B07MSPG7NJ4	-70.954357	0.0	70.00	9.00	
...	...	...	...	...	...	
91	B07R3ZFCZ48	-3.173328	0.0	373.78	130.00	
92	B07R4XJ3HY1	-1.536955	0.0	3407.55	19.00	
93	B07F5M62172	-1.050666	0.0	21534.67	2927.00	
94	B07R3ZKHYQ9	-0.954246	0.0	3196.88	278.63	
134	B07PHYB62D6	50.000000	NaN	30.00	0.00	

96 rows × 6 columns

It highlights ASINs with unuasual patters which have been detected as outlier due to (stockouts, big pricing changes, unusual traffic shift)

In [251…

```
# Exporting Outlier Data
outliers.to_csv('outliers_products.csv', index=False)
```

Insights

1. A small number of products are responsible for most of the 15 percent drop in organic conversion. For example, products like B07R3ZKHYQ9, B07R3TV8M37, and B07P6CBN1W6 have shown a large negative impact due to both lower traffic and lower conversion rates
2. Some products have organic traffic but zero units sold. This suggests possible stock out scenarios Examples include B07VQQ4QTM2 and B07MJBYM6K2
3. There is a significant decline in organic traffic for some high impact products. For example, B07R4V5W3K9 lost over 80 percent of its traffic compared to the previous month
4. There is no clear sign that major price changes are causing the conversion drop. The change in average selling price is minimal for most products
5. Paid ads have not offset the loss in organic traffic for many products. Products with high organic drops have not seen enough paid traffic to balance out the decline

Recommendations

1. Investigate possible stock out scenario for the products showing traffic but no sales. Fix listings or replenish inventory where needed
2. Work on improving organic rankings for the top affected products. Update product pages with better keywords, images, and titles. Offer discounts or coupons if needed to regain rank
3. Reassess ad campaigns for affected products. For products with large traffic loss and little ad support, consider increasing ad spend in the short term to protect market share
4. Monitor prices and ASP trends, but there is no immediate pricing action needed as price changes do not appear to be the main issue for the month of may
5. Focus on the top five to ten affected products first. Resolve inventory or listing issues within the next week to stabilize performance quickly