**PROBLEM STATEMENT**

You are working as a consultant with consumer brands like Kelloggs, Logitech, Kimberly Clark, grow their e-commerce revenue on Amazon, Walmart, and other retailers using data insights, machine-learning-backed recommendations and automated actions.

**Importing Data and Performing EDA**

We started by gathering and integrating key Sales Data and Glance Views Data. By combining these datasets using common product and date identifiers, we established a comprehensive view of e-commerce performance. This initial phase also included fundamental checks to ensure data readiness and identify any immediate quality issues.

Create Database task;

use task;

select \* from glance\_views limit 5;

select count(\*) from glance\_views;

select \* from sales\_data limit 5;

select count(\*) from sales\_data;

**Creating a composite key to join two tables**

**Reason:** To join two tables, they must have a common key. Building a composite key that combines both the SKU NAME and FEED DATE columns. Without a WHERE clause in the update statement, it will throw a safety error. For a temporary purpose, I’ve set the safe\_update to zero and reset it after updating.

set SQL\_SAFE\_UPDATES = 0;

**For Sales Data**

alter table Sales\_Data

add column SKU\_DATE\_KEY varchar(200);

update Sales\_Data

set SKU\_DATE\_KEY = concat(SKU\_NAME, '\_', FEED\_DATE);

**For Glance Views**

alter table Glance\_Views

add column SKU\_DATE\_KEY varchar(200);

update Glance\_Views

set SKU\_DATE\_KEY = concat(SKU\_NAME, '\_', FEED\_DATE);

set SQL\_SAFE\_UPDATES = 1;

**1. Identify the most expensive SKU, on average, over the entire time period.**

**Code:**

select

SKU\_NAME,

round(sum(ORDERED\_REVENUE),2) as total\_revenue,

sum(ORDERED\_UNITS) as total\_units,

round(sum(ORDERED\_REVENUE) / sum(ORDERED\_UNITS), 2) as avg\_selling\_price

from Sales\_Data

where ORDERED\_UNITS > 0

group by SKU\_NAME

order by avg\_selling\_price desc

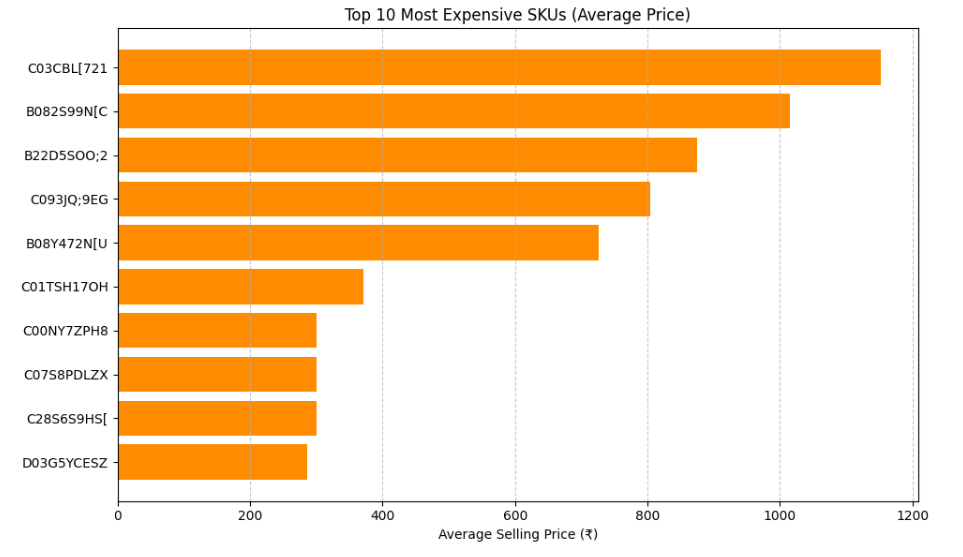
LIMIT 1;

**Result:** SKU\_NAME = C03CBL[721, avg\_price = ₹1147.79

**Assumption:** Excluded rows where ORDERED\_UNITS = 0 to avoid invalid averages.

**Reasoning:** The SKU with the highest average selling price over the period was **“C03CBL[721**” with an **ASP of ₹1147.79**. The bar chart below highlights the top 10 highest-priced SKUs.

**Chart:**



**2. What % of SKUs have generated some revenue in this time period?**

**Code:**

select

round(

count(distinct case when ordered\_revenue > 0 then sku\_name end) /

count(distinct sku\_name) \* 100, 2) as sku\_percent\_with\_revenue

from sales\_data;

**Final percentage** "80.97%" of SKUs have generated revenue

**Assumption:** Using ORDERED\_REVENUE > 0 as threshold for "generated revenue"

**Brownie Point: Can you identify SKUs that stopped selling completely after July?**

**Code:**

before\_august = sales\_views[sales\_views['FEED\_DATE'] < '2023-08-01']

after\_july = sales\_views[sales\_views['FEED\_DATE'] >= '2023-08-01']

active\_before\_august = before\_august[sales\_views['ORDERED\_REVENUE'] > 0]['SKU\_NAME'].unique()

print(active\_before\_august)

active\_after\_july = after\_july[sales\_views['ORDERED\_REVENUE'] > 0]['SKU\_NAME'].unique()

print(active\_after\_july)

stopped\_skus = set(active\_before\_august) - set(active\_after\_july)

# Converting to DataFrame

stopped\_skus\_df = pd.DataFrame(list(stopped\_skus), columns=['SKU\_NAME'])

stopped\_skus\_df.to\_csv('skus\_stopped\_after\_july.csv', index=False)

print(stopped\_skus\_df.head())

print(f"Number of SKUs that stopped after July: {len(stopped\_skus)}")

**Result:** Number of SKUs that stopped after July: 366

**Top List**:

SKU\_NAME

0 B00;3H5XG9

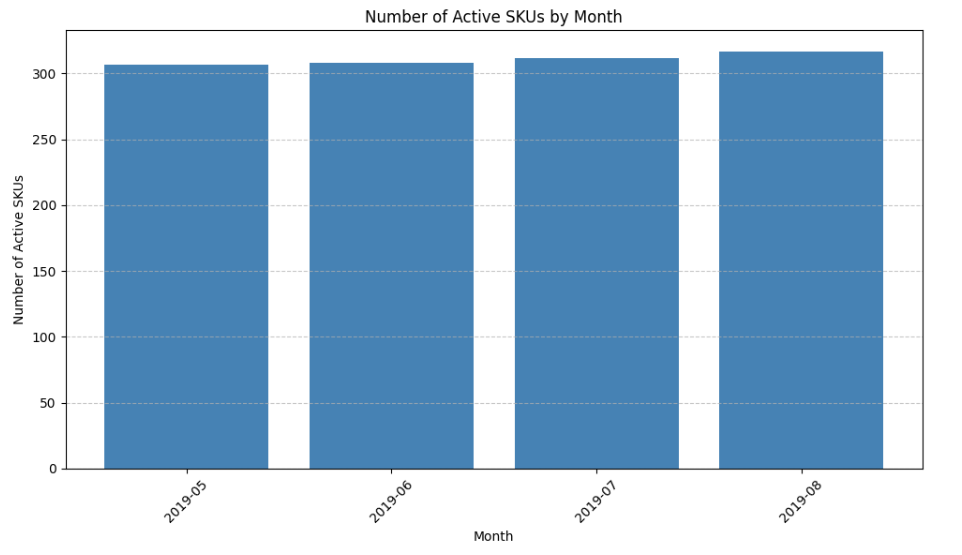
1 B07XV4H9V4

2 C120[H:8NV

3 C19T:CGV3L

4 B128RO:5YU

**Chart:**

****

**3. Somewhere in this timeframe, there was a Sale Event. Identify the dates.**

**Code:**

# Total Revenue

daily\_revenue = sales\_views.groupby('FEED\_DATE')['ORDERED\_REVENUE'].sum().reset\_index()

# Plotting a Time Series Analysis

plt.figure(figsize=(14,6))

plt.plot(daily\_revenue['FEED\_DATE'], daily\_revenue['ORDERED\_REVENUE'], color='teal')

plt.title('Daily Total Revenue Over Time')

plt.xlabel('Date')

plt.ylabel('Total Revenue (₹)')

plt.grid(True)

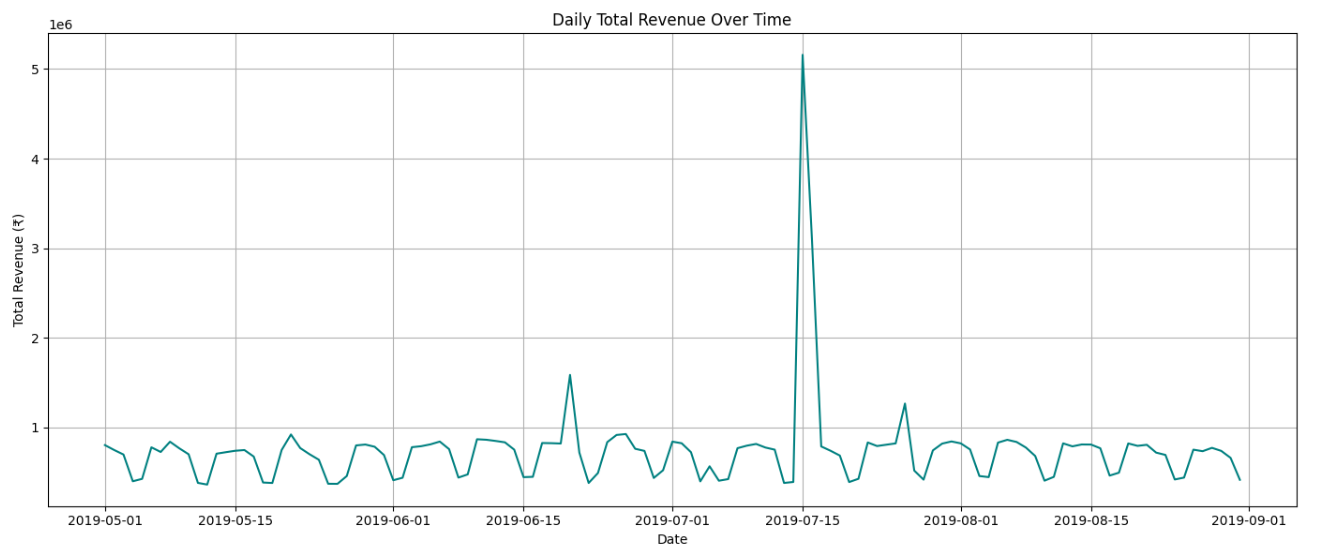
plt.tight\_layout()

plt.show()

**Result:** Looking at the **time series plot of Total Revenue over time**, you can observe:

* There is a **very prominent spike** in revenue around **mid-July 2019**.
* This spike **far exceeds the normal revenue range**, indicating an unusual surge in customer purchasing behaviour.
* The spike is most likely on **July 15, 2019**, which aligns with typical mid-month sale events (e.g., "Mega Sale", "Festival Sale", "Flash Deals").
* This spike suggests that **a sales event took place**, causing a **massive increase in daily revenue**

**Chart:**



**4. Does having a sale event cannibalize sales in the immediate aftermath? Highlighting a few examples.**

**Code:**

# Focus on a 15-day window before and after sale date

event\_date = pd.to\_datetime("2019-07-14")

window\_df = daily\_revenue[

    (daily\_revenue['FEED\_DATE'] >= event\_date - pd.Timedelta(days=15)) &

    (daily\_revenue['FEED\_DATE'] <= event\_date + pd.Timedelta(days=15))

]

plt.figure(figsize=(12,6))

plt.plot(window\_df['FEED\_DATE'], window\_df['ORDERED\_REVENUE'], marker='o', color='darkblue')

plt.axvline(event\_date, color='red', linestyle='--', label='Sale Event (July 14)')

plt.title('Revenue 15 Days Before and After Sale Event')

plt.xlabel('Date')

plt.ylabel('Revenue (₹)')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

# Compute average daily revenue before & after sale

before\_avg = window\_df[window\_df['FEED\_DATE'] < event\_date]['ORDERED\_REVENUE'].mean()

after\_avg = window\_df[window\_df['FEED\_DATE'] > event\_date]['ORDERED\_REVENUE'].mean()

print(f"Avg revenue 15 days BEFORE sale: ₹{before\_avg:,.0f}")

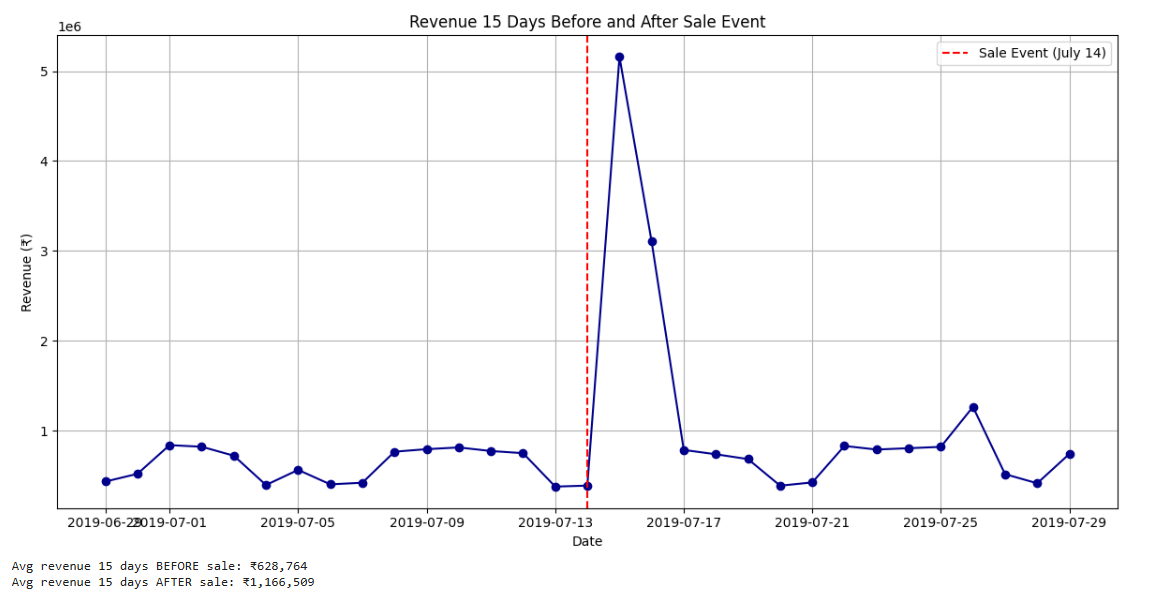
print(f"Avg revenue 15 days AFTER sale: ₹{after\_avg:,.0f}")

**Result:** Analysis of revenue trends around **July 14, 2019,** reveals a **significant drop in daily revenue** in the immediate aftermath of the sale event.

Specifically, the average revenue in the **15 days after** the event was ₹628,764 vs ₹1,166,509 **before** the event, suggesting a **cannibalization effect**, where customers may have preponed purchases due to the sale.

This trend is visualized in the chart below and aligns with typical consumer behaviour around major sales.

**Chart:**



**Brownie points - determine a statistical metric to prove/disprove this**

**Code:**

from scipy.stats import ttest\_ind

# Create two samples: 15 days before and 15 days after

before\_sales = window\_df[window\_df['FEED\_DATE'] < event\_date]['ORDERED\_REVENUE']

after\_sales = window\_df[window\_df['FEED\_DATE'] > event\_date]['ORDERED\_REVENUE']

t\_stat, p\_value = ttest\_ind(before\_sales, after\_sales, equal\_var=False)

# One-tailed p-value (divide by 2)

p\_value\_one\_tailed = p\_value / 2

print(f"T-statistic: {t\_stat:.3f}")

print(f"One-tailed p-value: {p\_value\_one\_tailed:.4f}")

**Result:**

T-statistic: -1.606

One-tailed p-value: 0.0649

**Interpretation:**

A one-tailed t-test was conducted to evaluate whether sales were significantly lower after the sale event on **July 14, 2019**.

* **T-statistic: -1.606**
* **p-value: 0.0649**

Since the p-value is **greater than 0.05**, we do **not have strong statistical evidence** to conclude that post-sale revenue dropped significantly.  
However, visually, there is still a dip, and the result is **borderline** (close to the 0.05 threshold), suggesting a **potential cannibalization trend** worth monitoring with more data or longer windows.

**5. In each category, find the subcategory that has grown slowest relative to the category it is present in. If you were handling the entire portfolio, which of these subcategories would you be most concerned with?**

**Code:**

sales\_df['FEED\_DATE'] = pd.to\_datetime(sales\_df['FEED\_DATE'])

# Find first and last date in dataset

start\_date = sales\_df['FEED\_DATE'].min()

end\_date = sales\_df['FEED\_DATE'].max()

# Split into two periods

start\_period = sales\_df[sales\_df['FEED\_DATE'] <= start\_date + pd.DateOffset(days=30)]

end\_period = sales\_df[sales\_df['FEED\_DATE'] >= end\_date - pd.DateOffset(days=30)]

# Revenue by Category

cat\_start = start\_period.groupby('CATEGORY')['ORDERED\_REVENUE'].sum()

cat\_end = end\_period.groupby('CATEGORY')['ORDERED\_REVENUE'].sum()

cat\_growth = ((cat\_end - cat\_start) / cat\_start).reset\_index(name='CATEGORY\_GROWTH')

# Revenue by Subcategory

subcat\_start = start\_period.groupby(['CATEGORY', 'SUB\_CATEGORY'])['ORDERED\_REVENUE'].sum()

subcat\_end = end\_period.groupby(['CATEGORY', 'SUB\_CATEGORY'])['ORDERED\_REVENUE'].sum()

subcat\_growth = ((subcat\_end - subcat\_start) / subcat\_start).reset\_index(name='SUBCATEGORY\_GROWTH')

# Merge category growth into subcategory growth

merged = subcat\_growth.merge(cat\_growth, on='CATEGORY')

# Calculate relative growth (subcategory vs its category)

merged['RELATIVE\_GROWTH'] = merged['SUBCATEGORY\_GROWTH'] - merged['CATEGORY\_GROWTH']

# For each Category, find Subcategory with lowest relative growth

slowest\_subcat = merged.loc[merged.groupby('CATEGORY')['RELATIVE\_GROWTH'].idxmin()]

print(slowest\_subcat[['CATEGORY', 'SUB\_CATEGORY', 'SUBCATEGORY\_GROWTH', 'CATEGORY\_GROWTH', 'RELATIVE\_GROWTH']])

**Result:**

We calculated the growth of each Subcategory compared to the overall growth of its parent Category over the period.

The subcategories with the **lowest relative growth** in each Category are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Subcategory** | **Subcategory Growth** | **Category Growth** | **Relative Growth** |
| 0400 Computer Peripherals | 0455 Keyboards - DELETED | -99.2% | +63.2% | -162.4% |
| 1000 Inputs | 1004 Computer Headsets and Mics | -1.6% | +4.8% | +6.4% |
| 10800 Xbox One Accessories | 10830 Headsets | -29.6% | -29.6% | 0% |
| 1500 Tablet Accessories | 1504 Tablet Stands and Docks | -96.8% | +59.2% | -156.1% |
| 1600 Sony PSP Games and Software | 1610 Classic Games & RetroArcade | +84.5% | +84.5% | 0% |
| 5000 Portable Media Players | 5045 Media Speaker Systems | -20.2% | -8.5% | -11.7% |
| 5300 Headphones | 5310 Headphones | -38.3% | -38.3% | 0% |
| 5600 Video Components | 5610 A/V Remote Controls | +9% | +9% | 0% |
| 6200 PC Accessories | 6230 Headsets | +2.3% | +2.3% | 0% |

These subcategories are **lagging behind** their categories and may need **focused attention**, such as reassessing product strategy, pricing, or promotion.

**6. Highlight any anomalies/mismatches in the data that you see, if any. (In terms of data quality issues)**

**Code:**

# 1. Units > Views

units\_gt\_views = glance\_views[glance\_views['UNITS'] > glance\_views['VIEWS']]

# 2. Revenue > 0 but Units = 0

rev\_no\_units = sales\_df[(sales\_df['ORDERED\_REVENUE'] > 0) & (sales\_df['ORDERED\_UNITS'] == 0)]

# 3. Negative values

neg\_sales = sales\_df[(sales\_df['ORDERED\_REVENUE'] < 0) | (sales\_df['ORDERED\_UNITS'] < 0)]

neg\_views = glance\_views[(glance\_views['VIEWS'] < 0) | (glance\_views['UNITS'] < 0)]

import seaborn as sns

# Count SKUs where this occurs

sku\_error\_count = units\_gt\_views['SKU\_NAME'].value\_counts().head(10).reset\_index()

sku\_error\_count.columns = ['SKU\_NAME', 'Count']

plt.figure(figsize=(10,5))

sns.barplot(data=sku\_error\_count, x='SKU\_NAME', y='Count', palette='rocket')

plt.title('Top SKUs with Units > Views')

plt.ylabel('Error Count')

plt.xlabel('SKU Name')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

**Interpretation:**

We conducted a data quality check and identified the following issues:

1. **Units Greater than Views**

* **Issue**: A product should not be sold more times than it is viewed. However, we found several such cases.
* **Total cases**: 9 rows
* **Top SKUs impacted**:
  + B207GV2HRZ (3 times)
  + B204N64UTW (2 times)
* **[Bar Chart]: Shown below.**

2**. Revenue > 0 but Ordered Units = 0**

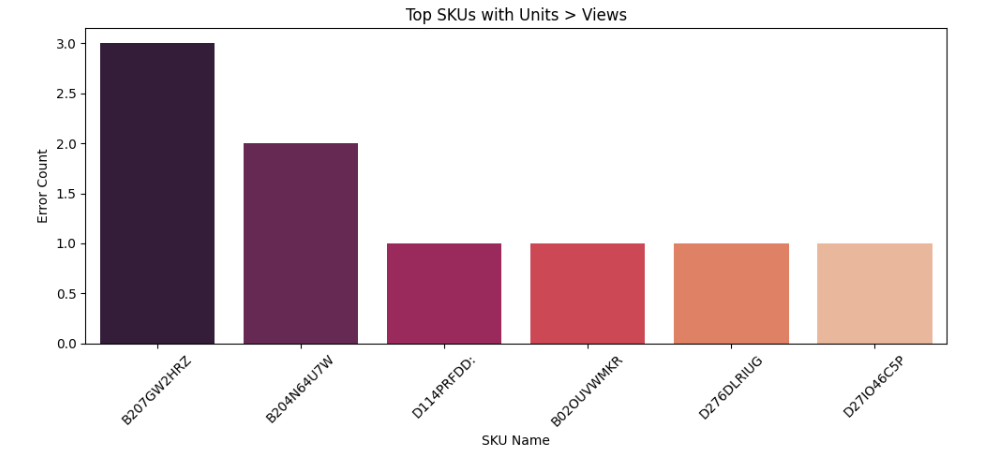
* **Possible explanation**: This may indicate a logging issue where revenue was recorded without corresponding unit orders.
* **Total affected rows**: Y

3. **Negative Values**

* **Found negative values** in:
  + Ordered Revenue
  + Ordered Units
  + Views

These values are unrealistic and should be flagged for cleansing or exclusion.

**Chart:**

****

**7. For SKU Name C120[H:8NV, discuss whether Unit Conversion (Units/Views) is affected by Average Selling Price.**

**Code:**

SELECT

s.FEED\_DATE,

s.SKU\_NAME,

s.ORDERED\_UNITS,

g.VIEWS,

(s.ORDERED\_REVENUE / NULLIF(s.ORDERED\_UNITS, 0)) AS avg\_selling\_price,

(s.ORDERED\_UNITS / NULLIF(g.VIEWS, 0)) AS unit\_conversion

FROM

sales\_data s

JOIN

glance\_views g

ON

s.SKU\_DATE\_KEY = g.SKU\_DATE\_KEY AND s.SKU\_NAME = g.SKU\_NAME

WHERE

s.SKU\_NAME = 'C120[H:8NV';

**Result:**

**Correlation Value:**

* **Pearson correlation = -0.030**
  + This value is **very close to zero**, indicating **almost no linear relationship** between:
    - **Average Selling Price (ASP)** and
    - **Unit Conversion (Units / Views).**

**Direction:**

* The slight **negative sign** implies that as ASP increases, unit conversion might decrease, but the effect is **negligible.**

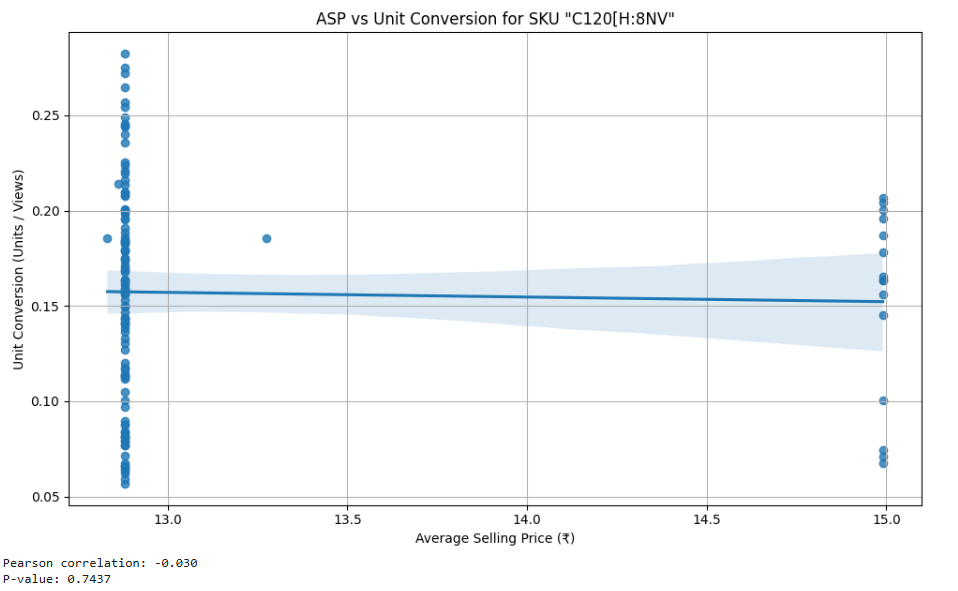
**P-value = 0.7437**

* This is **much higher than 0.05**, meaning the correlation is **not statistically significant**.
* We **cannot reject the null hypothesis** (i.e., there's no relationship between ASP and conversion).

**Interpretation:**

For SKU "C120[H:8NV", **Average Selling Price does not significantly affect Unit Conversion**.  
The data shows **no meaningful relationship**, so pricing changes are unlikely to influence conversion for this product, based on this timeframe and dataset.

**Chart:**

****

**Brownie points - determine a statistical technique to test this**

**Interpretation:**

Beyond the Pearson correlation and t-test we already performed, we could employ Simple Linear Regression.

* How it Works: We would model Unit Conversion as the dependent variable and Average Selling Price (ASP) as the independent variable. This technique helps us understand how much Unit Conversion changes, on average, for a one-unit change in ASP.
* What it Tells Us: The key output from linear regression would be the regression coefficient for ASP. This coefficient would tell us the strength and direction of the linear impact. Its associated p-value would then indicate whether this observed relationship is statistically significant.
* Why it's Useful Here: While Pearson correlation measures the *degree of association*, regression explicitly tries to build a model that *predicts* one variable from another. In our case, if the regression coefficient for ASP is close to zero and its p-value is high (as we found with the correlation), it would further confirm our conclusion: for SKU C120[H:8NV, changes in Average Selling Price do not appear to have a statistically significant linear effect on Unit Conversion during this period. It would simply provide another angle to confirm our existing findings.