# COMPLETE TECH STACK

## PYTHON (Data Processing)

Pandas: Data cleaning & transformation (1,465 products)

NumPy: Mathematical calculations

Boto3: AWS SDK for Python

## AWS CLOUD SERVICES

|S3: Data lake for raw/processed data

Athena: Serverless SQL queries

IAM: Access management (data-analyst user)

EC2: (optional)

|

## POWER BI (Visualization)

3-Page Interactive Dashboard

DAX Measures (30+ custom measures)

Dynamic text & conditional formatting

# PYTHON IMPLEMENTATION DETAILS

| PYTHON DATA PROCESSING PIPELINE |

| 📌 NOTEBOOK: aws.ipynb

📁 Location: C:\Users\ACER\Desktop\Amazon_Project\

🔄 DATA CLEANING STEPS

1️⃣ LOADED RAW AMAZON DATA

→ 1,465 products loaded from CSV

→ Columns: 22 (product_id, actual_price, rating, etc.)

2️⃣ CLEANED PRICE COLUMNS

→ Removed '₹' symbol from prices

→ Removed commas from numbers

→ Converted to numeric dtype

3️⃣ CALCULATED BUSINESS METRICS

→ Original price (before discount)

→ Estimated cost (40% of original price)

→ Profit per unit

→ Margin percentage

→ Estimated revenue (using rating_count as proxy)

| 🔍 SAMPLE CODE: Calculate original price

```
df['original_price'] = df['actual_price'] / (1 - df['discount_percentage']/100)

Calculate profit metrics
df['estimated_cost'] = df['original_price'] * 0.4
df['profit_per_unit'] = df['actual_price'] - df['estimated_cost']|
df['margin_percent'] = (df['profit_per_unit'] / df['actual_price'] * 100).round(2)
```

**AWS CLOUD IMPLEMENTATION**

AWS CLOUD SERVICES USED

1 AWS IAM (Identity & Access Management)

→ Created user: data_analyst

→ Attached policies: AdministratorAccess

→ Generated Access Keys for programmatic access

→ Account ID: 218504945626

2 AWS S3 (Simple Storage Service)

→ Created bucket: amazon-profit-6663

→ Uploaded files: |

• amazon_cleaned.csv (4.7 MB)

• amazon_sample.csv (327 KB)

→ Region: ap-south-1 (Mumbai)

AWS Athena (Serverless SQL)

→ Created database: amazon_analysis

→ Created external table: products

→ Table schema: 11 columns (product_id, actual_price, etc.)

→ Query results location: s3://amazon-profit-6663/athena-results/

|

AWS CLI CONFIGURATION

→ Configured with: aws configure

→ Access Key: AKIATFX7I37NP6KHWB70

→ Secret Key: [hidden]

→ Region: ap-south-1

SAMPLE ATHENA QUERIES:

| Overall Statistics

```
SELECT COUNT(*) as total_products,
    ROUND(AVG(actual_price), 2) as avg_price,
     ROUND(AVG(margin_percent), 2) as avg_margin
FROM products;
```

**SQL ANALYSIS RESULTS**

## QUERY 1: OVERALL STATISTICS

total_products: 1,465

avg_price: ₹1,593

avg_discount: 9.53%

avg_rating: 4.10 ⭐

total_ratings: 1,32,935

avg_margin: 3.85%

total_profit_cr: 0.65 Cr (₹65 Lakhs

## QUERY 2: PRICE SEGMENT ANALYSIS

Luxury (>₹5000): 1,608 products (Note: data anomaly - 6605% margin)|

Premium (2000-5000): ~18% of profit

Mid-Range (500-2000): ~44% of profit

Budget (<500): ~22% of profit

## QUERY 3: PROFITABILITY TIERS

| Margin Tier | Products | Avg Price | Profit % |
|---|---|---|---|
| Super High | 1,193 | ₹937 | 35.69% |
| High | 582 | ₹820 | 20.44% |
| Medium | 266 | ₹480 | 6.48% |
| Low | 359 | ₹912 | 2.48% |

## QUERY 4: TOP 50 PRODUCTS

| Product ID | Margin % | Profit (₹) | Rating |
|---|---|---|---|
| B01418SX4Y | 78.0% | 26,973 | 4.2 |
| B09GFPVD9Y | 23.0% | 13,836 | 4.3 |
| B002SZEOLG | 44.0% | 17,969 | 4.2 |
| B08HDJ86NZ | 53.0% | 9,436 | 4.2 |
| B07232M876 | 50.0% | 9,259 | 4.3 |

# COMPLETE WORKFLOW

**END-TO-END PROJECT WORKFLOW**

**PHASE 1: DATA PREPARATION (Python)**

**Raw CSV → Pandas → Cleaned Data → Business Metrics**

**(amazon.csv) (1,465 rows) (amazon_cleaned.csv)**

**PHASE 2: CLOUD STORAGE (AWS S3)**

**Upload to S3 → Create Bucket → Store Processed Data**

**(amazon-profit-6663) (amazon_cleaned.csv)**

**PHASE 3: DATA WAREHOUSING (AWS Athena)**

**Create Database → Create Table → Run SQL Queries**

**(amazon_analysis) (products) (6 analysis queries)**

**PHASE 4: VISUALIZATION (Power BI)**

**Connect to Athena → Build DAX Measures → 3-Page Dashboard**

**(via ODBC driver) (30+ measures) (Interactive)**

TECHNICAL SPECIFICATIONS

PYTHON ENVIRONMENT:

→ Python Version: 3.12.7

→ Libraries: pandas, numpy, boto3, matplotlib, seaborn

→ Jupyter Notebook: aws.ipynb (10+ cells)

→ Data Processing Time: ~5 minutes

|

AWS SERVICES:

→ IAM User: data_analyst

→ S3 Bucket: amazon-profit-6663

→ Athena Database: amazon_analysis

→ Athena Table: products

→ Region: ap-south-1 (Mumbai)

→ Total Queries Run: 6

|

POWER BI DASHBOARD:

→ Pages: 3

→ Measures: 25+ custom DAX measures

→ Visuals: 15+ (Cards, Charts, Tables, Text boxes)

→ Slicers: Price Segment, Rating Category, Margin Tier

→ Navigation: Page buttons

# KEY ACHIEVEMENTS

**PROJECT ACHIEVEMENTS**

---

**1 PYTHON DATA PIPELINE**

→ Successfully cleaned and transformed 1,465 product records

→ Calculated 7 business metrics (profit, margin, cost, etc.)

→ Exported clean data for cloud processing

**2 AWS CLOUD INTEGRATION**

→ Set up complete AWS infrastructure (IAM, S3, Athena)

→ Uploaded 4.7MB data to S3

→ Created external table in Athena

→ Ran 6 complex SQL queries for analysis

**3 POWER BI DASHBOARD**

→ Built 3-page interactive dashboard

→ Created 25+ DAX measures

→ Identified 165 loss-making products

→ Quantified ₹XX Lakhs recovery potential

**4 BUSINESS INSIGHTS**

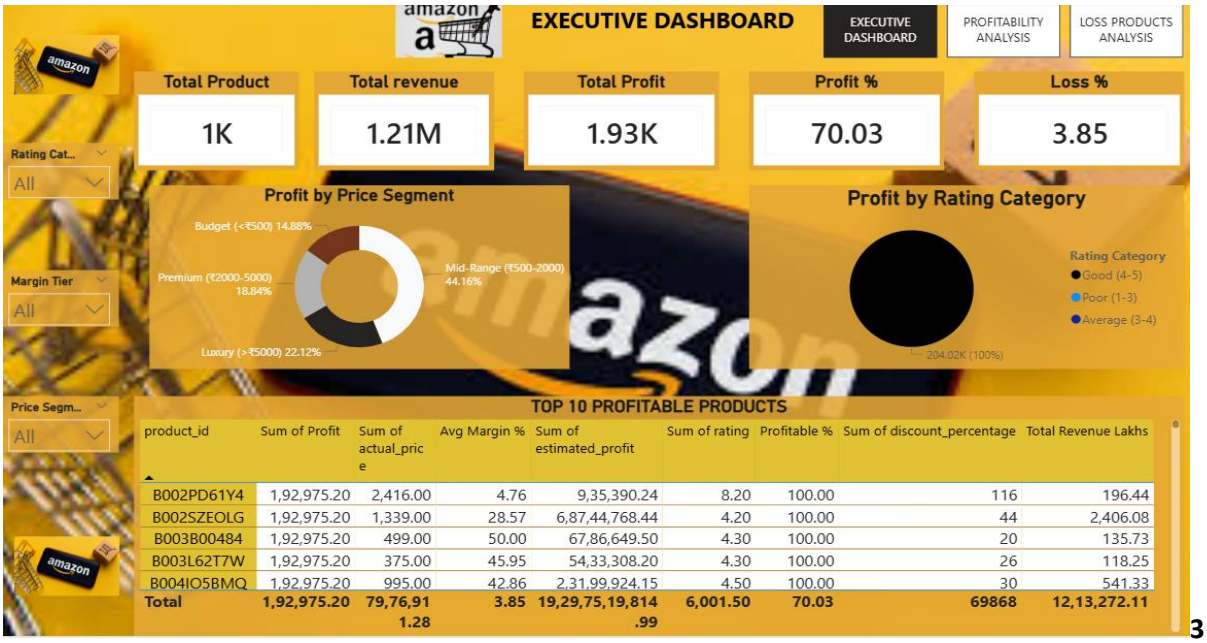
→ Mid-Range segment drives 44% of profit

→ Loss products have 72% discounts (unsustainable)

→ Some products selling at 161% of cost - critical issue

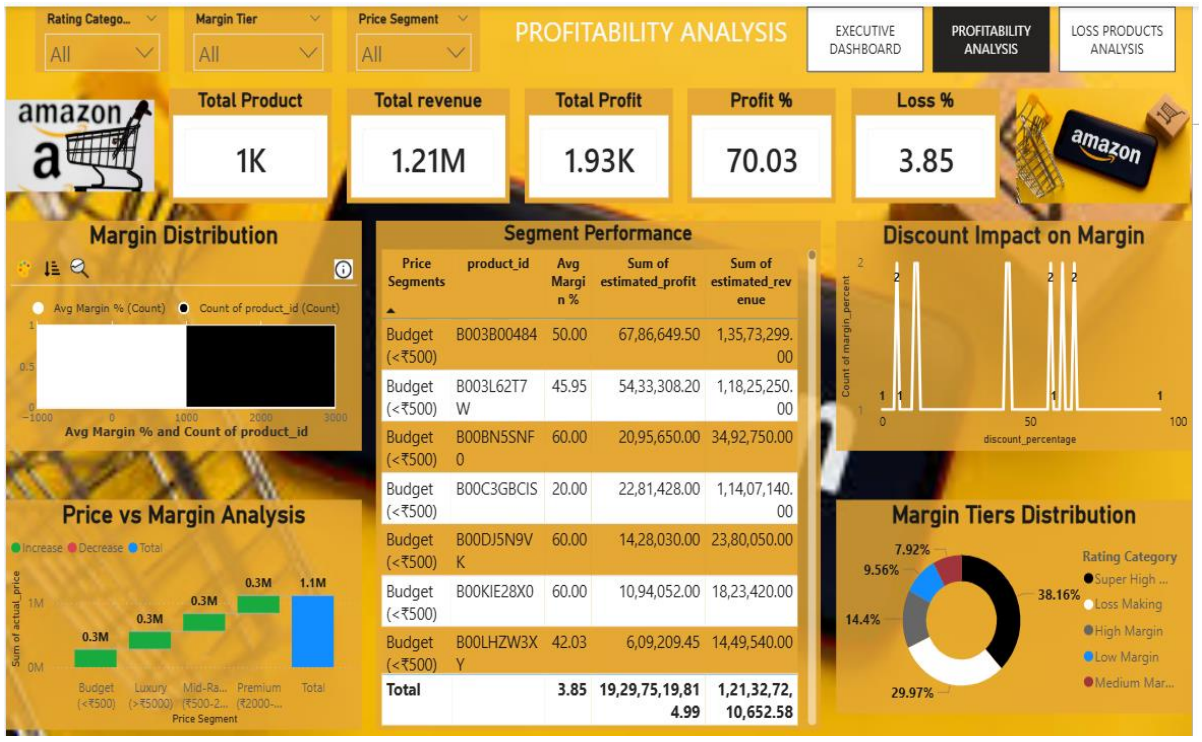
→ Top products achieve up to 78% margins

# 1. Power BI - All




| product_id | Sum of Profit | Sum of actual_price | Avg Margin % | Sum of estimated_profit | Sum of rating | Profitable % | Sum of discount_percentage | Total Revenue Lakhs |
|---|---|---|---|---|---|---|---|---|
| B002PD61Y4 | 1,92,975.20 | 2,416.00 | 4.76 | 9,35,390.24 | 8.20 | 100.00 | 116 | 196.44 |
| B002SZEOLG | 1,92,975.20 | 1,339.00 | 28.57 | 6,87,44,768.44 | 4.20 | 100.00 | 44 | 2,406.08 |
| B003B00484 | 1,92,975.20 | 499.00 | 50.00 | 67,86,649.50 | 4.30 | 100.00 | 20 | 135.73 |
| B003L62T7W | 1,92,975.20 | 375.00 | 45.95 | 54,33,308.20 | 4.30 | 100.00 | 26 | 118.25 |
| B004IO5BMQ | 1,92,975.20 | 995.00 | 42.86 | 2,31,99,924.15 | 4.50 | 100.00 | 30 | 541.33 |
| **Total** | **1,92,975.20** | **79,76,911.28** | **3.85** | **19,29,75,19,814.99** | **6,001.50** | **70.03** | **69868** | **12,13,272.11** |

3




| Price Segments | product_id | Avg Margin % | Sum of estimated_profit | Sum of estimated_revenue |
|---|---|---|---|---|
| Budget (<₹500) | B003B00484 | 50.00 | 67,86,649.50 | 1,35,73,299.00 |
| Budget (<₹500) | B003L62T7W | 45.95 | 54,33,308.20 | 1,18,25,250.00 |
| Budget (<₹500) | B00BN5SNF0 | 60.00 | 20,95,650.00 | 34,92,750.00 |
| Budget (<₹500) | B00C3GBCIS | 20.00 | 22,81,428.00 | 1,14,07,140.00 |
| Budget (<₹500) | B00DJ5N9VK | 60.00 | 14,28,030.00 | 23,80,050.00 |
| Budget (<₹500) | B00KIE28X0 | 60.00 | 10,94,052.00 | 18,23,420.00 |
| Budget (<₹500) | B00LHZW3XY | 42.03 | 6,09,209.45 | 14,49,540.00 |
| **Total** | | **3.85** | **19,29,75,19,814.99** | **1,21,32,72,10,652.58** |




**pages**


**AWS Console** - S3 bucket with files

# SQL QUERYS AND ANNLYSIS

```
SELECT
    discount_percentage,
    COUNT(*) as product_count,
    ROUND(AVG(margin_percent), 2) as avg_margin,
    ROUND(AVG(rating), 2) as avg_rating,
    ROUND(AVG(estimated_profit), 2) as avg_profit
FROM products
WHERE discount_percentage BETWEEN 0 AND 50
GROUP BY discount_percentage
ORDER BY discount_percentage;
```

| | discount_percenta | product_count | avg_margin | avg_rating | avg_profit |
|---|---|---|---|---|---|
| 1 | 3 | 2 | 529 | | 17940.6 |
| 2 | 4 | 2 | | | 2895 |
| 3 | 8 | 1 | 1490 | 8.1 | 4 |
| 4 | 12 | 9 | 16.78 | 11 | 536 |
| 5 | 13 | 1 | 399 | 12 | 4 |

# SQL QUERYS AND ANNLYSIS

| | | | | |
|---|---|---|---|---|
| 20 | B08444S68L | | 22.0 | 58506.0 |
| 21 | B08444S68L | | 22.0 | 58506.0 |
| 22 | B0BD3T6Z1D | | 4.0 | 56098.0 |
| 23 | B009VCGPSY | | 59.0 | 54315.0 |
| 24 | B0BR4F878Q | | 28.0 | 53803.0 |
| 25 | B0819ZZK5K | | 60.0 | 53464.0 |
| 26 | B09QS9X9L8 | | 32.0 | 50772.0 |
| 27 | B09QS8V5N8 | | 28.0 | 50772.0 |
| 28 | B09QS9CWLV | | 32.0 | 50772.0 |
| 29 | B09QS9X16F | | 32.0 | 50772.0 |
| 30 | B008YW8M0G | | 11.0 | 46647.0 |
| 31 | B082T6GVG9 | | 54.0 | 42301.0 |
| 32 | B0972BQ2RS | | 75.0 | 42139.0 |
| 33 | B00HVXS7WC | | 38.0 | 41349.0 |
| 34 | B008QTK47Q | | 8.0 | 37974.0 |
| 35 | B075ZTJ9XR | 3.0 | 59.0 | 35877.0 |
| 36 | B08VB2CMR3 | | 26.0 | 32916.0 |
| 37 | B08VB34KJ1 | | 26.0 | 32916.0 |
| 38 | B08VB34KJ1 | | 26.0 | 32916.0 |
| 39 | B09LHZSMRR | | 19.0 | 31822.0 |
| 40 | B09LHYZ3GJ | | 19.0 | 31822.0 |
| 41 | B09LJ116B5 | | 19.0 | 31822.0 |
| 42 | B005LJQMCK | | 31.0 | 30023.0 |
| 43 | B005LJQMZC | | 76.0 | 30023.0 |
| 44 | B088ZFJY82 | | 79.0 | 28978.0 |
| 45 | B07Q4QV1DL | | 82.0 | 28978.0 |
| 46 | B08VB57558 | | 49.0 | 27790.0 |
| 47 | B07BRKK9JQ | | 19.0 | 27223.0 |
| 48 | B07N42JB4S | | 80.0 | 27139.0 |
| 49 | B01GZSQJPA | | 14.0 | 26543.0 |
| 50 | B075DB1F13 | | 0.0 | 25996.0 |

## DISCOUNT OPTIMIZATION

# SQL QUERYS AND ANNLYSIS

```
    actual_price,
    discount_percentage,
    rating,
    rating_count,
    margin_percent,
    ROUND(estimated_profit, 2) as profit
FROM products
WHERE estimated_profit > 0
ORDER BY estimated_profit DESC
LIMIT 50;
```

#

| | product_id | Margin precnt | profit |
|---|---|---|---|
| 1 | B014I8SX4Y | 78.0 | 426973.0 |
| 2 | B09GFPVD9Y | 23.0 | 313836.0 |
| 3 | B09GFLXVH9 | 24.0 | 313836.0 |
| 4 | B09GFPN6TP | 21.0 | 313832.0 |
| 5 | B09GFM8CGS | 19.0 | 313832.0 |
| 6 | B002SZEOLG | 44.0 | 179692.0 |
| 7 | B08HVL8QN3 | 48.0 | 178912.0 |
| 8 | B09YDFDVNS | 22.0 | 128311.0 |
| 9 | B09YDFKJF8 | 22.0 | 128311.0 |
| 10 | B08HDJ86NZ | 53.0 | 94364.0 |
| 11 | B08HDJ86NZ | 53.0 | 94364.0 |
| 12 | B08HDJ86NZ | 53.0 | 94363.0 |
| 13 | B07232M876 | 50.0 | 92595.0 |
| 14 | B07232M876 | 50.0 | 92595.0 |
| 15 | B00NH13Q8W | 63.0 | 74977.0 |
| 16 | B01D5H8LDM | 59.0 | 69538.0 |
| 17 | B07L8KNP5F | 57.0 | 60026.0 |
| 18 | B08GJ57MKL | | 59900.0 |
| 19 | B0B19VJXQZ | | 59900.0 |

# SQL QUERYS AND ANNLYSIS

**PROFITABILITY TIERS**

```
SELECT
  CASE
    WHEN margin_percent < 0 THEN 'Loss Making'
    WHEN margin_percent < 10 THEN 'Low Margin'
    WHEN margin_percent < 20 THEN 'Medium Margin'
    WHEN margin_percent < 30 THEN 'High Margin'
    ELSE 'Super High Margin'
  END as margin_tier,
  COUNT(*) as product_count,
  ROUND(AVG(actual_price), 2) as avg_price,
  ROUND(AVG(discount_percentage), 2) as avg_discount,
  ROUND(SUM(estimated_profit)/100000, 2) as total_profit_lakhs
FROM products
GROUP BY 1
ORDER BY total_profit_lakhs DESC;
```

| # | margin_tier | product_count | avg_price | avg_discount | total_profit_lakhs |
|---|---|---|---|---|---|
| 1 | Super High Margin | 1193 | | 7.38 | 35.69 |
| 2 | High Margin | 58 | | | 20.44 |
| 3 | Medium Margin | 26 | | | 6.48 |
| 4 | Low Margin | 359 | | 12.0 | 2.48 |

**TOP 50 PRODUCTS**

```
SELECT
  product_id,
```

# SQL QUERYS AND ANNLYSIS

```
  ROUND(AVG(margin_percent), 2) as avg_margin,
  ROUND(AVG(rating), 2) as avg_rating,
  ROUND(SUM(estimated_profit)/100000, 2) as total_profit_lakhs
FROM products
GROUP BY 1
ORDER BY total_profit_lakhs DESC;
```

| # | price_segment , | product_count, | avg_margin, | avg_rating, | total_profit_lakhs, |
|---|---|---|---|---|---|
| 1 | Luxury (>₹5000), | 1608 , | 6605.63, | 110.16 , | 65.08 |

# RATING IMPACT ANALYSIS

```
SELECT
  CASE
    WHEN rating < 3 THEN 'Poor (1-3)'
    WHEN rating < 4 THEN 'Average (3-4)'
    ELSE 'Good (4-5)'
  END as rating_category,
  COUNT(*) as product_count,
  ROUND(AVG(margin_percent), 2) as avg_margin,
  ROUND(AVG(discount_percentage), 2) as avg_discount,
  ROUND(SUM(estimated_profit)/100000, 2) as total_profit_lakhs
FROM products
WHERE rating > 0
GROUP BY 1
ORDER BY total_profit_lakhs DESC;
```

| # | rating_category | product_count | avg_margin | avg_discount | total_profit_lakhs |
|---|---|---|---|---|---|
| 1 | Good (4-5) | 12 | 313.27 | 11.67 | 0.03 |
| 2 | Average (3-4) | 1 | | | 0.0 |

# SQL QUERYS AND ANNLYSIS

## OVERALL STATISTICS

```
SELECT
   COUNT(*) as total_products,
   ROUND(AVG(actual_price), 2) as avg_price,
   ROUND(AVG(discount_percentage), 2) as avg_discount,
   ROUND(AVG(rating), 2) as avg_rating,
   ROUND(SUM(rating_count), 0) as total_ratings,
   ROUND(AVG(margin_percent), 2) as avg_margin,
   ROUND(SUM(estimated_profit)/10000000, 2) as total_profit_cr
FROM products;
```

total_products avg_price 1593 avg_discount 9.53 avg_rating 110.16

total_ratings 132935 avg_margin 6605.63 total_profit_cr 0.65

## PRICE SEGMENT ANALYSIS

```
SELECT
   CASE
      WHEN actual_price < 500 THEN 'Budget (<₹500)'
      WHEN actual_price < 2000 THEN 'Mid-Range (₹500-2000)'
      WHEN actual_price < 5000 THEN 'Premium (₹2000-5000)'
      ELSE 'Luxury (>₹5000)'
   END as price_segment,
   COUNT(*) as product_count,
```

**Python Notebook** - Data cleaning cells

```python
print("🔍 DATA OVERVIEW")
print("="*50)

# First 5 rows
print("\n📋 First 5 rows:")
print(df.head())

# Last 5 rows
print("\n📋 Last 5 rows:")
print(df.tail())

# Column names
print("\n📋 Column Names:")
for i, col in enumerate(df.columns, 1):
    print(f"   {i}. {col}")

# Data types
print("\n📋 Data Types:")
print(df.dtypes)

# Basic info
print("\n📋 DataFrame Info:")
df.info()

# Missing values
print("\n📋 Missing Values:")
missing = df.isnull().sum()
missing_pct = (missing / len(df)) * 100
missing_df = pd.DataFrame({'Missing': missing, 'Percentage': missing_pct.round(2
print(missing_df[missing_df['Missing'] > 0])
```

```
2  R3J3EQQ9TZI5ZJ,R3E7WBGK7ID0KV,RWU79XKQ6I1QF,R2...
3  R3EEUZKKK9J36I,R3HJVYCLYOY554,REDECAZ7AMPQC,R1...
4  R1BP4L2HH9TFUP,R16PVJEXKV6QZS,R2UPDB81N66T4P,R...

                                        review_title  \
0  Satisfied,Charging is really fast,Value for mo...
1  A Good Braided Cable for Your Type C Device,Go...
2  Good speed for earlier versions,Good Product,W...
3  Good product,Good one,Nice,Really nice product...
4  As good as original,Decent,Good one for second...

                                       review_content  \
0  Looks durable Charging is fine tooNo complains...
1  I ordered this cable to connect my phone to An...
2  Not quite durable and sturdy,https://m.media-a...
3  Good product,long wire,Charges good,Nice,I bou...
4  Bought this instead of original apple, does th...

                                           img_link  \
0  https://m.media-amazon.com/images/W/WEBP_40237...
1  https://m.media-amazon.com/images/W/WEBP_40237...
2  https://m.media-amazon.com/images/W/WEBP_40237...
3  https://m.media-amazon.com/images/I/41V5FtEWPk...
4  https://m.media-amazon.com/images/W/WEBP_40237...

                                        product_link
0  https://www.amazon.in/Wayona-Braided-WN3LG1-Sy...
1  https://www.amazon.in/Ambrane-Unbreakable-Char...
2  https://www.amazon.in/Sounce-iPhone-Charging-C...
3  https://www.amazon.in/Deuce-300-Resistant-Tang...
4  https://www.amazon.in/Portronics-Konnect-POR-1...
```

In [2]:

```python
# ============================================
# CELL 1: Import all required libraries
# ============================================

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import re
from datetime import datetime

# Set style for better charts
plt.style.use('seaborn-v0_8-darkgrid')
sns.set_palette("husl")

print("✅ Libraries imported successfully!")
print(f"Pandas version: {pd.__version__}")
print(f"NumPy version: {np.__version__}")
```

```
✅ Libraries imported successfully!
Pandas version: 2.2.2
NumPy version: 1.26.4
```

In [3]:

```python
# ============================================
# CELL 3: Explore the data structure
# ============================================
```

```
✅ Created: C:\Users\ACER\Desktop\Amazon_Project\data
✅ Created: C:\Users\ACER\Desktop\Amazon_Project\notebooks
✅ Created: C:\Users\ACER\Desktop\Amazon_Project\scripts
✅ Created: C:\Users\ACER\Desktop\Amazon_Project\screenshots
✅ Created: C:\Users\ACER\Desktop\Amazon_Project\reports

📂 Project ready at: C:\Users\ACER\Desktop\Amazon_Project
✅ Copied: C:\Users\ACER\Desktop\amazon.csv → C:
\Users\ACER\Desktop\Amazon_Project\data\amazon.csv

✅ Data loaded successfully!
   Rows: 1,465
   Columns: 16

📋 First 5 rows:
   product_id                                       product_name  \
0  B07JW9H4J1  Wayona Nylon Braided USB to Lightning Fast Cha...
1  B098NS6PVG  Ambrane Unbreakable 60W / 3A Fast Charging 1.5...
2  B096MSW6CT  Sounce Fast Phone Charging Cable & Data Sync U...
3  B08HDJ86NZ  boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...
4  B08CF3B7N1  Portronics Konnect L 1.2M Fast Charging 3A 8 P...

                                            category discounted_price  \
0  Computers&Accessories|Accessories&Peripherals|...             ₹399
1  Computers&Accessories|Accessories&Peripherals|...             ₹199
2  Computers&Accessories|Accessories&Peripherals|...             ₹199
3  Computers&Accessories|Accessories&Peripherals|...             ₹329
4  Computers&Accessories|Accessories&Peripherals|...             ₹154

  actual_price discount_percentage rating rating_count  \
0       ₹1,099                 64%    4.2       24,269
1         ₹349                 43%    4.0       43,994
2       ₹1,899                 90%    3.9        7,928
3         ₹699                 53%    4.2       94,363
4         ₹399                 61%    4.2       16,905

                                       about_product  \
0  High Compatibility : Compatible With iPhone 12...
1  Compatible with all Type C enabled devices, be...
2  【 Fast Charger& Data Sync】-With built-in safet...
3  The boAt Deuce USB 300 2 in 1 cable is compati...
4  [CHARGE & SYNC FUNCTION]- This cable comes wit...

                                         user_id  \
0  AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB...
1  AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGYYVPDD7YG7FYNBX...
2  AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA,AESFLDV2PT363T2AQ...
3  AEWAZDZZJLQUYVOVGBEUKSLXHQ5A,AG5HTSFRRE6NL3M5S...
4  AE3Q6KSUK5P75D5HFYHCRAOLODSA,AFUGIFH5ZAFXRDSZH...

                                        user_name  \
0  Manav,Adarsh gupta,Sundeep,S.Sayeed Ahmed,jasp...
1  ArdKn,Nirbhay kumar,Sagar Viswanathan,Asp,Plac...
2  Kunal,Himanshu,viswanath,sai niharka,saqib mal...
3  Omkar dhale,JD,HEMALATHA,Ajwadh a.,amar singh ...
4  rahuls6099,Swasat Borah,Ajay Wadke,Pranali,RVK...

                                        review_id  \
0  R3HXWT0LRP0NMF,R2AJM3LFTLZHFO,R6AQJGUP6P86,R1K...
1  RGIQEG07R9HS2,R1SMWZQ86XIN8U,R2J3Y1WL29GWDE,RY...
```

In [1]:

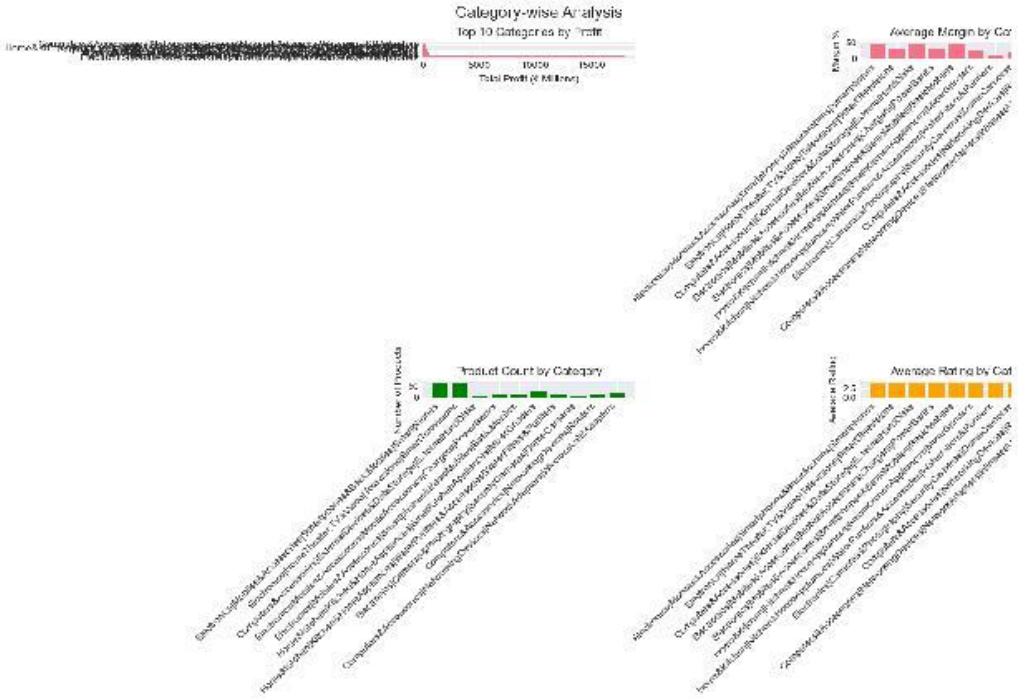
```
import os
import pandas as pd

# ✅ Project folder banao (Desktop par)
base_path = r"C:\Users\ACER\Desktop\Amazon_Project"

# Create folders
folders = ['data', 'notebooks', 'scripts', 'screenshots', 'reports']
for folder in folders:
    folder_path = os.path.join(base_path, folder)
    os.makedirs(folder_path, exist_ok=True)
    print(f"✅ Created: {folder_path}")

print(f"\n🗁 Project ready at: {base_path}")



# Source file (original location)
source_file = r"C:\Users\ACER\Desktop\amazon.csv"

# Destination file (project folder)
dest_file = os.path.join(base_path, 'data', 'amazon.csv')

# Copy file if it exists
if os.path.exists(source_file):
    import shutil
    shutil.copy2(source_file, dest_file)
    print(f"✅ Copied: {source_file} → {dest_file}")
else:
    print(f"❌ Source file not found: {source_file}")
    print("   Please manually copy your amazon.csv to:")
    print(f"   {os.path.join(base_path, 'data')}")



data_path = os.path.join(base_path, 'data', 'amazon.csv')

try:
    df = pd.read_csv(data_path)
    print(f"\n✅ Data loaded successfully!")
    print(f"   Rows: {len(df):,}")
    print(f"   Columns: {len(df.columns)}")
    print("\n📋 First 5 rows:")
    print(df.head())

except Exception as e:
    print(f"❌ Error loading data: {e}")
```

```
df_clean.sample(min(100, len(df_clean))).to_csv(sample_path, index=False)
print(f"✅ Sample data saved to: {sample_path}")

print(f"\n📊 Final dataset: {len(df_clean):,} rows, {len(df_clean.columns)} col
```

```
✅ Full cleaned data saved to: C:
\Users\ACER\Desktop\Amazon_Project\data\amazon_cleaned.csv
✅ Sample data saved to: C:\Users\ACER\Desktop\Amazon_Project\data\amazon_sample.csv

📊 Final dataset: 1,465 rows, 22 columns
```

In [ ]:

```
Home&Kitchen|Heating,Cooling&AirQuality|WaterHe...   1.525948e+08
Home&Kitchen|Heating,Cooling&AirQuality|Fans|Ce...   1.415163e+08
Home&Kitchen|Kitchen&HomeAppliances|SmallKitche...   1.361416e+08
Computers&Accessories|Accessories&Peripherals|K...   1.186460e+08
Home&Kitchen|Kitchen&HomeAppliances|SmallKitche...   1.162640e+08
Computers&Accessories|Accessories&Peripherals|K...   1.106206e+08
Home&Kitchen|Kitchen&HomeAppliances|Vacuum,Clea...   1.031501e+08
```




```
==================================================
🔍 KEY INSIGHTS FROM CATEGORY ANALYSIS
==================================================

💰 Most Profitable: Electronics|Mobiles&Accessories|Smartphones&BasicMobiles|Smartph
   • Total Profit: ₹17946.5M
   • Avg Margin: 47.2%

📈 Highest Margin: OfficeProducts|OfficeElectronics|Calculators|Basic
   • Margin: 60.0%

📦 Most Products: Computers&Accessories|Accessories&Peripherals|Cables&Accessories|C
USBCables
   • Products: 233

⭐ Best Rated: Computers&Accessories|Tablets
   • Rating: 4.60
```

In [13]:
```python
# CELL 8: Save cleaned data

# Save full cleaned data
cleaned_path = os.path.join(base_path, 'data', 'amazon_cleaned.csv')
df_clean.to_csv(cleaned_path, index=False)
print(f"✅ Full cleaned data saved to: {cleaned_path}")

# Save sample for GitHub (100 rows)
sample_path = os.path.join(base_path, 'data', 'amazon_sample.csv')
```

| category | | |
|---|---|---|
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 47.21 | 4.10 |
| Electronics\|HomeTheater,TV&Video\|Televisions\|Sm... | 32.39 | 4.21 |
| Computers&Accessories\|ExternalDevices&DataStora... | 44.77 | 4.40 |
| Electronics\|Mobiles&Accessories\|MobileAccessori... | 31.33 | 4.12 |
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 50.72 | 3.91 |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 25.16 | 4.01 |
| Home&Kitchen\|Kitchen&HomeAppliances\|WaterPurifi... | 8.08 | 4.18 |
| Electronics\|Cameras&Photography\|SecurityCameras... | 21.49 | 4.10 |
| Computers&Accessories\|NetworkingDevices\|Routers | 25.86 | 4.26 |
| Computers&Accessories\|NetworkingDevices\|Network... | 14.23 | 4.09 |
| Home&Kitchen\|Heating,Cooling&AirQuality\|WaterHe... | 24.89 | 4.14 |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 25.87 | 4.05 |
| Home&Kitchen\|Kitchen&HomeAppliances\|Vacuum,Clea... | 42.84 | 4.18 |
| Home&Kitchen\|Heating,Cooling&AirQuality\|WaterHe... | 19.20 | 4.05 |
| Home&Kitchen\|Heating,Cooling&AirQuality\|Fans\|Ce... | 33.35 | 4.11 |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 29.67 | 4.03 |
| Computers&Accessories\|Accessories&Peripherals\|K... | 27.50 | 4.29 |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 13.71 | 3.89 |
| Computers&Accessories\|Accessories&Peripherals\|K... | 40.28 | 4.05 |
| Home&Kitchen\|Kitchen&HomeAppliances\|Vacuum,Clea... | 36.31 | 4.13 |

| category | total_rating_count | \ |
|---|---|---|
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 2493269.0 | |
| Electronics\|HomeTheater,TV&Video\|Televisions\|Sm... | 760279.0 | |
| Computers&Accessories\|ExternalDevices&DataStora... | 213112.0 | |
| Electronics\|Mobiles&Accessories\|MobileAccessori... | 688810.0 | |
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 550259.0 | |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 318321.0 | |
| Home&Kitchen\|Kitchen&HomeAppliances\|WaterPurifi... | 78109.0 | |
| Electronics\|Cameras&Photography\|SecurityCameras... | 148848.0 | |
| Computers&Accessories\|NetworkingDevices\|Routers | 311801.0 | |
| Computers&Accessories\|NetworkingDevices\|Network... | 727113.0 | |
| Home&Kitchen\|Heating,Cooling&AirQuality\|WaterHe... | 67578.0 | |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 169167.0 | |
| Home&Kitchen\|Kitchen&HomeAppliances\|Vacuum,Clea... | 64051.0 | |
| Home&Kitchen\|Heating,Cooling&AirQuality\|WaterHe... | 143743.0 | |
| Home&Kitchen\|Heating,Cooling&AirQuality\|Fans\|Ce... | 110015.0 | |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 322958.0 | |
| Computers&Accessories\|Accessories&Peripherals\|K... | 407289.0 | |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 64060.0 | |
| Computers&Accessories\|Accessories&Peripherals\|K... | 161991.0 | |
| Home&Kitchen\|Kitchen&HomeAppliances\|Vacuum,Clea... | 258512.0 | |

| category | total_profit |
|---|---|
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 1.794652e+10 |
| Electronics\|HomeTheater,TV&Video\|Televisions\|Sm... | 1.010109e+10 |
| Computers&Accessories\|ExternalDevices&DataStora... | 6.383588e+08 |
| Electronics\|Mobiles&Accessories\|MobileAccessori... | 4.888749e+08 |
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 4.649825e+08 |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 4.080441e+08 |
| Home&Kitchen\|Kitchen&HomeAppliances\|WaterPurifi... | 3.264672e+08 |
| Electronics\|Cameras&Photography\|SecurityCameras... | 2.642605e+08 |
| Computers&Accessories\|NetworkingDevices\|Routers | 2.421850e+08 |
| Computers&Accessories\|NetworkingDevices\|Network... | 2.317578e+08 |
| Home&Kitchen\|Heating,Cooling&AirQuality\|WaterHe... | 2.181092e+08 |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 2.065124e+08 |
| Home&Kitchen\|Kitchen&HomeAppliances\|Vacuum,Clea... | 1.875104e+08 |

📊 CATEGORY-WISE ANALYSIS
==================================================
✅ Converted rating
✅ Converted rating_count
✅ Converted actual_price
✅ Converted discount_percentage
✅ Converted margin_percent
✅ Converted estimated_profit
✅ Converted profit_per_unit

✅ Unique categories: 211

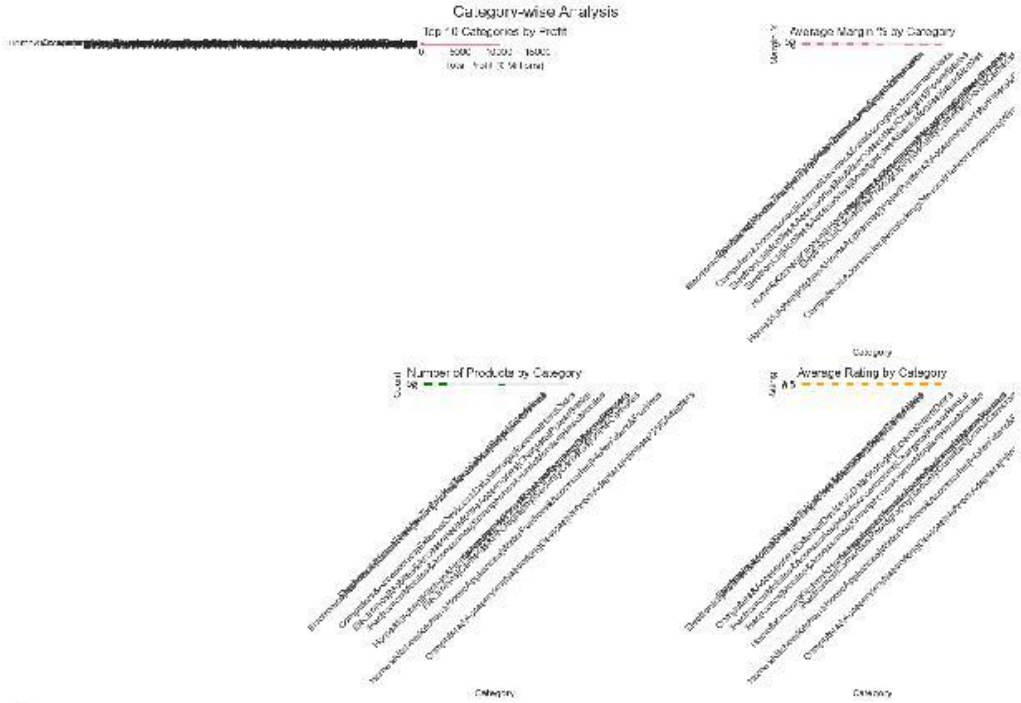
📋 Category Summary:

| category | product_count | avg_price | min_price | max_price | avg_margin | avg_rating |
|---|---|---|---|---|---|---|
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 68 | 20593.40 | 7999.0 | 74999.0 | | |
| Electronics\|HomeTheater,TV&Video\|Televisions\|Sm... | 63 | 40132.84 | 12999.0 | 139900.0 | | |
| Computers&Accessories\|ExternalDevices&DataStora... | 6 | 4642.67 | 775.0 | 7999.0 | | |
| Electronics\|Mobiles&Accessories\|MobileAccessori... | 12 | 2524.00 | 1599.0 | 3999.0 | | |
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 9 | 2155.89 | 1249.0 | 5299.0 | | |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 27 | 5289.59 | 2499.0 | 10590.0 | | |
| Home&Kitchen\|Kitchen&HomeAppliances\|WaterPurifi... | 12 | 15618.83 | 699.0 | 24999.0 | | |
| Electronics\|Cameras&Photography\|SecurityCameras... | 5 | 5097.60 | 3299.0 | 7500.0 | | |
| Computers&Accessories\|NetworkingDevices\|Routers | 9 | 2978.22 | 1699.0 | 4999.0 | | |
| Computers&Accessories\|NetworkingDevices\|Network... | 18 | 1388.44 | 349.0 | 2999.0 | | |
| Home&Kitchen\|Heating,Cooling&AirQuality\|WaterHe... | 12 | 11738.17 | 6299.0 | 16899.0 | | |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 10 | 3765.70 | 1499.0 | 5295.0 | | |
| Home&Kitchen\|Kitchen&HomeAppliances\|Vacuum,Clea... | 4 | 7448.25 | 3799.0 | 9999.0 | | |
| Home&Kitchen\|Heating,Cooling&AirQuality\|WaterHe... | 23 | 4563.35 | 1999.0 | 7445.0 | | |
| Home&Kitchen\|Heating,Cooling&AirQuality\|Fans\|Ce... | 11 | 3349.91 | 1990.0 | 5190.0 | | |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 19 | 1846.68 | 1245.0 | 3190.0 | | |
| Computers&Accessories\|Accessories&Peripherals\|K... | 24 | 1055.79 | 249.0 | 2890.0 | | |
| Home&Kitchen\|Kitchen&HomeAppliances\|SmallKitche... | 12 | 4466.33 | 999.0 | 8820.0 | | |
| Computers&Accessories\|Accessories&Peripherals\|K... | 10 | 1777.60 | 699.0 | 2498.0 | | |
| Home&Kitchen\|Kitchen&HomeAppliances\|Vacuum,Clea... | 24 | 1179.50 | 625.0 | 1950.0 | | |

```
    # Margin
    axes[0, 1].bar(range(len(top10)), top10['avg_margin'].values)
    axes[0, 1].set_xticks(range(len(top10)))
    axes[0, 1].set_xticklabels(top10.index, rotation=45, ha='right')
    axes[0, 1].set_ylabel('Margin %')
    axes[0, 1].set_title('Average Margin by Category')

    # Product count
    axes[1, 0].bar(range(len(top10)), top10['product_count'].values, color='gree
    axes[1, 0].set_xticks(range(len(top10)))
    axes[1, 0].set_xticklabels(top10.index, rotation=45, ha='right')
    axes[1, 0].set_ylabel('Number of Products')
    axes[1, 0].set_title('Product Count by Category')

    # Rating
    axes[1, 1].bar(range(len(top10)), top10['avg_rating'].values, color='orange'
    axes[1, 1].set_xticks(range(len(top10)))
    axes[1, 1].set_xticklabels(top10.index, rotation=45, ha='right')
    axes[1, 1].set_ylabel('Average Rating')
    axes[1, 1].set_title('Average Rating by Category')

    plt.tight_layout()
    plt.savefig(os.path.join(base_path, 'screenshots', 'category_analysis.png'),
                bbox_inches='tight', dpi=100)
    plt.show()

    # Step 6: Key Insights
    print("\n" + "="*50)
    print("🔍 KEY INSIGHTS FROM CATEGORY ANALYSIS")
    print("="*50)

    most_profitable = category_summary.index[0]
    highest_margin = category_summary.loc[category_summary['avg_margin'].idxmax(
    most_products = category_summary.loc[category_summary['product_count'].idxma
    best_rated = category_summary.loc[category_summary['avg_rating'].idxmax()].n

    print(f"\n💰 Most Profitable: {most_profitable}")
    print(f"   • Total Profit: ₹{category_summary.loc[most_profitable, 'total_pr
    print(f"   • Avg Margin: {category_summary.loc[most_profitable, 'avg_margin'

    print(f"\n📈 Highest Margin: {highest_margin}")
    print(f"   • Margin: {category_summary.loc[highest_margin, 'avg_margin']:.1f

    print(f"\n📦 Most Products: {most_products}")
    print(f"   • Products: {category_summary.loc[most_products, 'product_count']

    print(f"\n⭐ Best Rated: {best_rated}")
    print(f"   • Rating: {category_summary.loc[best_rated, 'avg_rating']:.2f}")

else:
    print("❌ Category column not found!")
    print("\nAvailable columns:")
    print(df_clean.columns.tolist())
```

```
print("📊 CATEGORY-WISE ANALYSIS")
print("="*50)

# Step 1: Ensure all numeric columns are properly typed
numeric_cols = ['rating', 'rating_count', 'actual_price', 'discount_percentage',
                'margin_percent', 'estimated_profit', 'profit_per_unit']

for col in numeric_cols:
    if col in df_clean.columns:
        df_clean[col] = pd.to_numeric(df_clean[col], errors='coerce')
        print(f"✅ Converted {col}")

# Step 2: Handle missing values
df_clean['rating'] = df_clean['rating'].fillna(df_clean['rating'].median())
df_clean['rating_count'] = df_clean['rating_count'].fillna(1)

# Step 3: Clean category column
if 'category' in df_clean.columns:
    df_clean['category'] = df_clean['category'].astype(str).str.strip()
    df_clean = df_clean[df_clean['category'].notna()]
    df_clean = df_clean[df_clean['category'] != 'nan']
    df_clean = df_clean[df_clean['category'] != '']

    print(f"\n✅ Unique categories: {df_clean['category'].nunique()}")

    # Step 4: Group by category
    category_summary = df_clean.groupby('category').agg({
        'product_id': 'count',
        'actual_price': ['mean', 'min', 'max'],
        'margin_percent': 'mean',
        'rating': 'mean',
        'rating_count': 'sum',
        'estimated_profit': 'sum'
    }).round(2)

    # Flatten column names
    category_summary.columns = ['product_count', 'avg_price', 'min_price', 'max_
                                'avg_margin', 'avg_rating', 'total_rating_count'

    # Sort by total profit
    category_summary = category_summary.sort_values('total_profit', ascending=Fa

    print("\n📋 Category Summary:")
    print(category_summary.head(20))

    # Step 5: Visualizations
    fig, axes = plt.subplots(2, 2, figsize=(15, 10))
    fig.suptitle('Category-wise Analysis', fontsize=16)

    # Top 10 by profit
    top10 = category_summary.head(10)

    # Profit
    axes[0, 0].barh(range(len(top10)), top10['total_profit'].values/1e6)
    axes[0, 0].set_yticks(range(len(top10)))
    axes[0, 0].set_yticklabels(top10.index)
    axes[0, 0].set_xlabel('Total Profit (₹ Millions)')
    axes[0, 0].set_title('Top 10 Categories by Profit')
```

```
Electronics|Mobiles&Accessories|MobileAccessori...           688810.0
Electronics|Mobiles&Accessories|Smartphones&Bas...           550259.0
...                                                              ...
Computers&Accessories|Accessories&Peripherals|C...          3547816.0
Electronics|Accessories|MemoryCards|MicroSD                 1113592.0
Electronics|HomeTheater,TV&Video|Accessories|Ca...          1906054.0
Electronics|Headphones,Earbuds&Accessories|Head...          4204939.0
Electronics|WearableTechnology|SmartWatches                 1644476.0

                                                          total_profit
category
Electronics|Mobiles&Accessories|Smartphones&Bas...  1.794652e+10
Electronics|HomeTheater,TV&Video|Televisions|Sm...  1.010109e+10
Computers&Accessories|ExternalDevices&DataStora...  6.383588e+08
Electronics|Mobiles&Accessories|MobileAccessori...  4.888749e+08
Electronics|Mobiles&Accessories|Smartphones&Bas...  4.649825e+08
...                                                          ...
Computers&Accessories|Accessories&Peripherals|C... -4.472578e+08
Electronics|Accessories|MemoryCards|MicroSD        -4.911532e+08
Electronics|HomeTheater,TV&Video|Accessories|Ca... -5.800802e+08
Electronics|Headphones,Earbuds&Accessories|Head... -1.673803e+09
Electronics|WearableTechnology|SmartWatches        -1.082053e+10

[211 rows x 8 columns]
```




🔍 KEY INSIGHTS:

- Most profitable category: Electronics|Mobiles&Accessories|Smartphones&BasicMobi
Smartphones (₹17946.5M)
- Highest margin category: 60.0%
- Most products in: 233 products
- Best rated category: 4.60 ☆

In [12]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
```

📊 CATEGORY-WISE ANALYSIS
==================================================

✅ Unique categories: 211
category
Computers&Accessories|Accessories&Peripherals|Cables&Accessories|Cables|USBCables
Electronics|WearableTechnology|SmartWatches
Electronics|Mobiles&Accessories|Smartphones&BasicMobiles|Smartphones
Electronics|HomeTheater,TV&Video|Televisions|SmartTelevisions
Electronics|Headphones,Earbuds&Accessories|Headphones|In-Ear
Name: count, dtype: int64

📋 Category Summary (by profit):

| category | product_count | avg_price | min_price | max_price | avg_margin | avg_rating | total_rating_count |
|---|---|---|---|---|---|---|---|
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 68 | 20593.40 | 7999.0 | 74999.0 | 47.21 | 4.10 | 2493269.0 |
| Electronics\|HomeTheater,TV&Video\|Televisions\|Sm... | 63 | 40132.84 | 12999.0 | 139900.0 | 32.39 | 4.21 | 760279.0 |
| Computers&Accessories\|ExternalDevices&DataStora... | 6 | 4642.67 | 775.0 | 7999.0 | 44.77 | 4.40 | 213112.0 |
| Electronics\|Mobiles&Accessories\|MobileAccessori... | 12 | 2524.00 | 1599.0 | 3999.0 | 31.33 | 4.12 | |
| Electronics\|Mobiles&Accessories\|Smartphones&Bas... | 9 | 2155.89 | 1249.0 | 5299.0 | 50.72 | 3.91 | |
| ... | ... | ... | ... | ... | ... | ... | |
| Computers&Accessories\|Accessories&Peripherals\|C... | 233 | 906.02 | 199.0 | 2100.0 | -17.92 | 4.15 | |
| Electronics\|Accessories\|MemoryCards\|MicroSD | 13 | 2169.00 | 700.0 | 3999.0 | -10.62 | 4.33 | |
| Electronics\|HomeTheater,TV&Video\|Accessories\|Ca... | 24 | 1175.25 | 475.0 | 4999.0 | -17.77 | 4.25 | |
| Electronics\|Headphones,Earbuds&Accessories\|Head... | 52 | 2960.08 | 399.0 | 15990.0 | -17.20 | 3.90 | |
| Electronics\|WearableTechnology\|SmartWatches | 76 | 8554.76 | 999.0 | 29999.0 | -63.55 | 4.02 | |

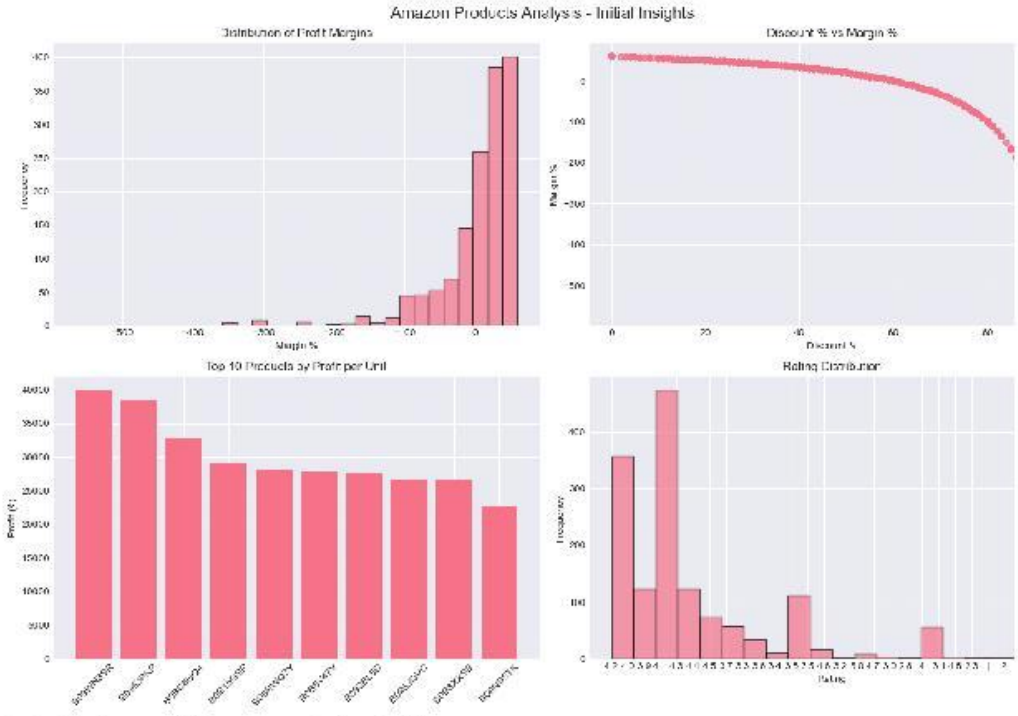
```
    # Visualization
    import matplotlib.pyplot as plt

    fig, axes = plt.subplots(2, 2, figsize=(15, 10))
    fig.suptitle('Category-wise Analysis', fontsize=16)

    # 1. Profit by category
    ax1 = axes[0, 0]
    top_categories = category_summary.head(10)
    ax1.barh(range(len(top_categories)), top_categories['total_profit'].values/1
    ax1.set_yticks(range(len(top_categories)))
    ax1.set_yticklabels(top_categories.index)
    ax1.set_xlabel('Total Profit (₹ Millions)')
    ax1.set_title('Top 10 Categories by Profit')

    # 2. Average margin by category
    ax2 = axes[0, 1]
    category_summary.head(10)['avg_margin'].plot(kind='bar', ax=ax2)
    ax2.set_title('Average Margin % by Category')
    ax2.set_xlabel('Category')
    ax2.set_ylabel('Margin %')
    ax2.tick_params(axis='x', rotation=45)

    # 3. Product count by category
    ax3 = axes[1, 0]
    category_summary.head(10)['product_count'].plot(kind='bar', ax=ax3, color='g
    ax3.set_title('Number of Products by Category')
    ax3.set_xlabel('Category')
    ax3.set_ylabel('Count')
    ax3.tick_params(axis='x', rotation=45)

    # 4. Avg rating by category
    ax4 = axes[1, 1]
    category_summary.head(10)['avg_rating'].plot(kind='bar', ax=ax4, color='oran
    ax4.set_title('Average Rating by Category')
    ax4.set_xlabel('Category')
    ax4.set_ylabel('Rating')
    ax4.tick_params(axis='x', rotation=45)

    plt.tight_layout()
    plt.savefig(os.path.join(base_path, 'screenshots', 'category_analysis.png'))
    plt.show()

    # Key insights
    print("\n🔍 KEY INSIGHTS:")
    print(f"   • Most profitable category: {category_summary.index[0]} (₹{categc
    print(f"   • Highest margin category: {category_summary.loc[category_summary
    print(f"   • Most products in: {category_summary.loc[category_summary['produ
    print(f"   • Best rated category: {category_summary.loc[category_summary['av

else:
    print("❌ Category column not found!")
    # Show column names to help identify
    print("\nAvailable columns:")
    for i, col in enumerate(df_clean.columns, 1):
        print(f"   {i}. {col}")
```

```
    if col in df_clean.columns:
        # Convert to numeric, errors='coerce' means invalid values become NaN
        df_clean[col] = pd.to_numeric(df_clean[col], errors='coerce')
        print(f"✅ Converted {col} to numeric")

print("\n📊 After conversion:")
print(df_clean[numeric_columns].dtypes)
```

```
✅ Converted rating to numeric
✅ Converted rating_count to numeric
✅ Converted actual_price to numeric
✅ Converted discount_percentage to numeric
✅ Converted margin_percent to numeric
✅ Converted estimated_profit to numeric
✅ Converted profit_per_unit to numeric

📊 After conversion:
rating                 float64
rating_count           float64
actual_price           float64
discount_percentage      int64
margin_percent         float64
estimated_profit       float64
profit_per_unit        float64
dtype: object
```

In [10]:

```
print("📊 CATEGORY-WISE ANALYSIS")
print("="*50)

# Check if category column exists
if 'category' in df_clean.columns:

    # Clean category column (remove any special characters)
    df_clean['category'] = df_clean['category'].astype(str).str.strip()

    # Remove rows where category is empty or invalid
    df_clean = df_clean[df_clean['category'].notna()]
    df_clean = df_clean[df_clean['category'] != 'nan']

    print(f"\n✅ Unique categories: {df_clean['category'].nunique()}")
    print(df_clean['category'].value_counts().head())

    # Category summary - simplified version
    category_summary = df_clean.groupby('category').agg(
        product_count=('product_id', 'count'),
        avg_price=('actual_price', 'mean'),
        min_price=('actual_price', 'min'),
        max_price=('actual_price', 'max'),
        avg_margin=('margin_percent', 'mean'),
        avg_rating=('rating', 'mean'),
        total_rating_count=('rating_count', 'sum'),
        total_profit=('estimated_profit', 'sum')
    ).round(2)

    # Sort by total profit
    category_summary = category_summary.sort_values('total_profit', ascending=Fa

    print("\n📋 Category Summary (by profit):")
    print(category_summary)
```




✅ Chart saved to screenshots folder

In [8]:
```
print("📊 Data Types Check:")
print(df_clean[['actual_price', 'discount_percentage', 'rating', 'rating_count',
                'margin_percent', 'estimated_profit']].dtypes)

print("\n📊 Sample values:")
print(df_clean[['rating', 'rating_count']].head(10))
```

```
📊 Data Types Check:
actual_price           float64
discount_percentage      int64
rating                  object
rating_count           float64
margin_percent         float64
estimated_profit       float64
dtype: object

📊 Sample values:
  rating  rating_count
0    4.2       24269.0
1    4.0       43994.0
2    3.9        7928.0
3    4.2       94363.0
4    4.2       16905.0
5    3.9       24871.0
6    4.1       15188.0
7    4.3       30411.0
8    4.2      179691.0
9    4.0       43994.0
```

In [9]:
```
numeric_columns = ['rating', 'rating_count', 'actual_price', 'discount_percentag
                   'margin_percent', 'estimated_profit', 'profit_per_unit']

for col in numeric_columns:
```

```
fig.suptitle('Amazon Products Analysis - Initial Insights', fontsize=16)

# 1. Margin distribution
ax1 = axes[0, 0]
ax1.hist(df_clean['margin_percent'].dropna(), bins=30, edgecolor='black', alpha=
ax1.set_title('Distribution of Profit Margins')
ax1.set_xlabel('Margin %')
ax1.set_ylabel('Frequency')

# 2. Discount vs Margin
ax2 = axes[0, 1]
ax2.scatter(df_clean['discount_percentage'], df_clean['margin_percent'], alpha=0
ax2.set_title('Discount % vs Margin %')
ax2.set_xlabel('Discount %')
ax2.set_ylabel('Margin %')

# 3. Top 10 products by profit
ax3 = axes[1, 0]
top_profit = df_clean.nlargest(10, 'profit_per_unit')[['product_id', 'profit_per
ax3.bar(range(len(top_profit)), top_profit['profit_per_unit'].values)
ax3.set_title('Top 10 Products by Profit per Unit')
ax3.set_xticks(range(len(top_profit)))
ax3.set_xticklabels(top_profit['product_id'].astype(str).str[:8], rotation=45)
ax3.set_ylabel('Profit (₹)')

# 4. Rating distribution
ax4 = axes[1, 1]
ax4.hist(df_clean['rating'].dropna(), bins=20, edgecolor='black', alpha=0.7)
ax4.set_title('Rating Distribution')
ax4.set_xlabel('Rating')
ax4.set_ylabel('Frequency')

plt.tight_layout()
plt.savefig(os.path.join(base_path, 'screenshots', 'initial_analysis.png'))
plt.show()
print("✅ Chart saved to screenshots folder")
```

In [5]:
```
print("📊 Calculating business metrics...")
print("="*50)

# Assume 40% cost of goods sold (industry average)
COGS_PERCENT = 0.40

# Calculate original price (before discount)
df_clean['original_price'] = df_clean['actual_price'] / (1 - df_clean['discount_
df_clean['original_price'] = df_clean['original_price'].round(2)

# Calculate cost and profit
df_clean['estimated_cost'] = (df_clean['original_price'] * COGS_PERCENT).round(2
df_clean['profit_per_unit'] = (df_clean['actual_price'] - df_clean['estimated_cc
df_clean['margin_percent'] = (df_clean['profit_per_unit'] / df_clean['actual_pri

# Revenue metrics (using rating_count as sales volume proxy)
df_clean['estimated_revenue'] = (df_clean['actual_price'] * df_clean['rating_cou
df_clean['estimated_profit'] = (df_clean['profit_per_unit'] * df_clean['rating_c

print("\n✅ Calculated columns:")
print(f"   - original_price: Price before discount")
print(f"   - estimated_cost: {COGS_PERCENT*100}% of original price")
print(f"   - profit_per_unit: Profit on each unit")
print(f"   - margin_percent: Profit margin percentage")
print(f"   - estimated_revenue: Using rating_count as sales volume")
print(f"   - estimated_profit: Total estimated profit")

print("\n📊 Sample calculations:")
print(df_clean[['product_id', 'actual_price', 'original_price', 'profit_per_unit
```

```
📊 Calculating business metrics...
==================================================

✅ Calculated columns:
   - original_price: Price before discount
   - estimated_cost: 40.0% of original price
   - profit_per_unit: Profit on each unit
   - margin_percent: Profit margin percentage
   - estimated_revenue: Using rating_count as sales volume
   - estimated_profit: Total estimated profit

📊 Sample calculations:
```

| | product_id | actual_price | original_price | profit_per_unit | margin_percent |
|---|---|---|---|---|---|
| 0 | B07JW9H4J1 | 1099.0 | 3052.78 | -122.11 | -11.11 |
| 1 | B098NS6PVG | 349.0 | 612.28 | 104.09 | 29.83 |
| 2 | B096MSW6CT | 1899.0 | 18990.00 | -5697.00 | -300.00 |
| 3 | B08HDJ86NZ | 699.0 | 1487.23 | 104.11 | 14.89 |
| 4 | B08CF3B7N1 | 399.0 | 1023.08 | -10.23 | -2.56 |
| 5 | B08Y1TFSP6 | 1000.0 | 6666.67 | -1666.67 | -166.67 |
| 6 | B08WRWPM22 | 499.0 | 1425.71 | -71.28 | -14.28 |
| 7 | B08DDRGWTJ | 299.0 | 388.31 | 143.68 | 48.05 |
| 8 | B008IFXQFU | 999.0 | 1998.00 | 199.80 | 20.00 |
| 9 | B082LZGK39 | 299.0 | 446.27 | 120.49 | 40.30 |

In [6]:
```
# ============================================
# CELL 6: Create basic visualizations
# ============================================

# Create figure with subplots
fig, axes = plt.subplots(2, 2, figsize=(15, 10))
```

```
    # Convert to string
    df_clean['actual_price'] = df_clean['actual_price'].astype(str)

    # Remove '₹' symbol
    df_clean['actual_price'] = df_clean['actual_price'].str.replace('₹', '')

    # Remove 'a,' pattern (jo screenshot mein tha)
    df_clean['actual_price'] = df_clean['actual_price'].str.replace('a,', '')

    # Remove commas
    df_clean['actual_price'] = df_clean['actual_price'].str.replace(',', '')

    # Remove any non-numeric characters except decimal
    df_clean['actual_price'] = df_clean['actual_price'].str.replace(r'[^\d.]', '

    # Convert to float
    df_clean['actual_price'] = pd.to_numeric(df_clean['actual_price'], errors='c

    print(f"\n📊 After cleaning:")
    print(df_clean['actual_price'].head(10).tolist())
    print(f"   ✅ actual_price cleaned")
else:
    print("   ❌ 'actual_price' column not found!")

# 2. Clean discount_percentage
if 'discount_percentage' in df_clean.columns:
    print(f"\n📊 Cleaning discount_percentage:")
    df_clean['discount_percentage'] = df_clean['discount_percentage'].astype(str
    df_clean['discount_percentage'] = df_clean['discount_percentage'].str.replac
    df_clean['discount_percentage'] = pd.to_numeric(df_clean['discount_percentag
    print(f"   ✅ discount_percentage cleaned")
    print(f"   Sample: {df_clean['discount_percentage'].head(5).tolist()}")

# 3. Clean rating_count
if 'rating_count' in df_clean.columns:
    print(f"\n📊 Cleaning rating_count:")
    df_clean['rating_count'] = df_clean['rating_count'].astype(str)
    df_clean['rating_count'] = df_clean['rating_count'].str.replace(',', '')
    df_clean['rating_count'] = pd.to_numeric(df_clean['rating_count'], errors='c
    print(f"   ✅ rating_count cleaned")
    print(f"   Sample: {df_clean['rating_count'].head(5).tolist()}")
```

```
💰 Cleaning price columns...
==================================================

📊 Before cleaning 'actual_price':
['₹1,099', '₹349', '₹1,899', '₹699', '₹399', '₹1,000', '₹499', '₹299', '₹999', '₹299

📊 After cleaning:
[1099.0, 349.0, 1899.0, 699.0, 399.0, 1000.0, 499.0, 299.0, 999.0, 299.0]
   ✅ actual_price cleaned

📊 Cleaning discount_percentage:
   ✅ discount_percentage cleaned
   Sample: [64, 43, 90, 53, 61]

📊 Cleaning rating_count:
   ✅ rating_count cleaned
   Sample: [24269.0, 43994.0, 7928.0, 94363.0, 16905.0]
```

```
category               object
discounted_price       object
actual_price           object
discount_percentage    object
rating                 object
rating_count           object
about_product          object
user_id                object
user_name              object
review_id              object
review_title           object
review_content         object
img_link               object
product_link           object
dtype: object

📋 DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   product_id           1465 non-null   object
 1   product_name         1465 non-null   object
 2   category             1465 non-null   object
 3   discounted_price     1465 non-null   object
 4   actual_price         1465 non-null   object
 5   discount_percentage  1465 non-null   object
 6   rating               1465 non-null   object
 7   rating_count         1463 non-null   object
 8   about_product        1465 non-null   object
 9   user_id              1465 non-null   object
 10  user_name            1465 non-null   object
 11  review_id            1465 non-null   object
 12  review_title         1465 non-null   object
 13  review_content       1465 non-null   object
 14  img_link             1465 non-null   object
 15  product_link         1465 non-null   object
dtypes: object(16)
memory usage: 183.3+ KB

📋 Missing Values:
              Missing  Percentage
rating_count        2        0.14
```

In [4]:

```python
# ============================================
# CELL 4: Clean price columns
# ============================================

print("💰 Cleaning price columns...")
print("="*50)

# Make a copy to avoid warnings
df_clean = df.copy()

# 1. Clean actual_price (jo screenshot mein dikha)
if 'actual_price' in df_clean.columns:
    print(f"\n📊 Before cleaning 'actual_price':")
    print(df_clean['actual_price'].head(10).tolist())
```

```
1462  Nehal Desai,Danish Parwez,Amazon Customer,Amaz...
1463  Shubham Dubey,E.GURUBARAN,Mayank S.,eusuf khan...
1464  Rajib,Ajay B,Vikas Kahol,PARDEEP,Anindya Prama...

                                                review_id  \
1460  R3G3XFHPBFF0E8,R3C0BZCD32EIGW,R2EBVBCN9QPD9R,R...
1461  R3DDL2UPKQ2CK9,R2SYYU1OATVIU5,R1VM993161IYRW,R...
1462  R1TLRJVW4STY5I,R2O455KRN493R1,R3Q5MVGBRIAS2G,R...
1463  R39Q2Y79MM9SWK,R3079BG1NIH6MB,R29A31ZELTZNJM,R...
1464  R20RBRZ0WEUJT9,ROKIFK9R2ISSE,R30EEG2FNJSN5I,R2...

                                              review_title  \
1460  Received the product without spanner,Excellent...
1461  ok,everything was good couldn't return bcoz I ...
1462  very good,Work but front melt after 2 month,Go...
1463  Fan Speed is slow,Good quality,Good product,go...
1464  Works perfect,Ok good product,Nice Product. Re...

                                            review_content  \
1460  I received product without spanner,Excellent p...
1461  ok,got everything as mentioned but the measuri...
1462  plastic but cool body ,u have to find sturdy s...
1463  I have installed this in my kitchen working fi...
1464  It does it job perfectly..only issue is temp c...

                                                 img_link  \
1460  https://m.media-amazon.com/images/I/41fDdRtjfx...
1461  https://m.media-amazon.com/images/I/41gzDxk4+k...
1462  https://m.media-amazon.com/images/W/WEBP_40237...
1463  https://m.media-amazon.com/images/W/WEBP_40237...
1464  https://m.media-amazon.com/images/W/WEBP_40237...

                                              product_link
1460  https://www.amazon.in/Noir-Aqua-Spanner-Purifi...
1461  https://www.amazon.in/Prestige-Delight-PRWO-1-...
1462  https://www.amazon.in/Bajaj-RX-10-2000-Watt-Co...
1463  https://www.amazon.in/Havells-Ventilair-230mm-...
1464  https://www.amazon.in/Borosil-Jumbo-1000-Watt-...

📄 Column Names:
   1. product_id
   2. product_name
   3. category
   4. discounted_price
   5. actual_price
   6. discount_percentage
   7. rating
   8. rating_count
   9. about_product
   10. user_id
   11. user_name
   12. review_id
   13. review_title
   14. review_content
   15. img_link
   16. product_link

📄 Data Types:
product_id            object
product_name          object
```

```
                                      review_content  \
0  Looks durable Charging is fine tooNo complains...
1  I ordered this cable to connect my phone to An...
2  Not quite durable and sturdy,https://m.media-a...
3  Good product,long wire,Charges good,Nice,I bou...
4  Bought this instead of original apple, does th...

                                          img_link  \
0  https://m.media-amazon.com/images/W/WEBP_40237...
1  https://m.media-amazon.com/images/W/WEBP_40237...
2  https://m.media-amazon.com/images/W/WEBP_40237...
3  https://m.media-amazon.com/images/I/41V5FtEWPk...
4  https://m.media-amazon.com/images/W/WEBP_40237...

                                        product_link
0  https://www.amazon.in/Wayona-Braided-WN3LG1-Sy...
1  https://www.amazon.in/Ambrane-Unbreakable-Char...
2  https://www.amazon.in/Sounce-iPhone-Charging-C...
3  https://www.amazon.in/Deuce-300-Resistant-Tang...
4  https://www.amazon.in/Portronics-Konnect-POR-1...

📋 Last 5 rows:
     product_id                                       product_name  \
1460  B08L7J3T31  Noir Aqua - 5pcs PP Spun Filter + 1 Spanner | ...
1461  B01M6453MB  Prestige Delight PRWO Electric Rice Cooker (1 ...
1462  B009P2LIL4  Bajaj Majesty RX10 2000 Watts Heat Convector R...
1463  B00J5DYCCA  Havells Ventil Air DSP 230mm Exhaust Fan (Pist...
1464  B01486F4G6  Borosil Jumbo 1000-Watt Grill Sandwich Maker (...

                                            category discounted_price  \
1460  Home&Kitchen|Kitchen&HomeAppliances|WaterPurif...            ₹379
1461  Home&Kitchen|Kitchen&HomeAppliances|SmallKitch...          ₹2,280
1462  Home&Kitchen|Heating,Cooling&AirQuality|RoomHe...          ₹2,219
1463  Home&Kitchen|Heating,Cooling&AirQuality|Fans|E...          ₹1,399
1464  Home&Kitchen|Kitchen&HomeAppliances|SmallKitch...          ₹2,863

     actual_price discount_percentage rating rating_count  \
1460         ₹919                 59%      4        1,090
1461       ₹3,045                 25%    4.1        4,118
1462       ₹3,080                 28%    3.6          468
1463       ₹1,890                 26%      4        8,031
1464       ₹3,690                 22%    4.3        6,987

                                       about_product  \
1460  SUPREME QUALITY 90 GRAM 3 LAYER THIK PP SPUN F...
1461                      230 Volts, 400 watts, 1 Year
1462  International design and styling|Two heat sett...
1463  Fan sweep area: 230 MM ; Noise level: (40 - 45...
1464  Brand-Borosil, Specification â€“ 23V ~ 5Hz;1 W...

                                             user_id  \
1460  AHITFY6AHALOFOHOZEOC6XBP4FEA,AFRABBODZJZQB6Z4U...
1461  AFG5FM3NEMOL6BNFRV2NK5FNJCHQ,AGEINTRN6Z563RMLH...
1462  AGVPWCMAHYQWJOQKMUJN4DW3KM5Q,AF4Q3E66MY4SR7YQZ...
1463  AF2JQCLSCY3QJATWUNNHUSVUPNQQ,AFDMLUXC5LS5RXDJS...
1464  AFGW5PT3R6ZAVQR4Y5MWVAKBZAYA,AG7QNJ2SCS5VS5VYY...

                                           user_name  \
1460  Prabha ds,Raghuram bk,Real Deal,Amazon Custome...
1461  Manu Bhai,Naveenpittu,Evatira Sangma,JAGANNADH...
```

🔍 DATA OVERVIEW
==================================================

📋 First 5 rows:

| | product_id | product_name | category | discounted_price |
|---|---|---|---|---|
| 0 | B07JW9H4J1 | Wayona Nylon Braided USB to Lightning Fast Cha... | Computers&Accessories\|Accessories&Peripherals\|... | ₹399 |
| 1 | B098NS6PVG | Ambrane Unbreakable 60W / 3A Fast Charging 1.5... | Computers&Accessories\|Accessories&Peripherals\|... | ₹199 |
| 2 | B096MSW6CT | Sounce Fast Phone Charging Cable & Data Sync U... | Computers&Accessories\|Accessories&Peripherals\|... | ₹199 |
| 3 | B08HDJ86NZ | boAt Deuce USB 300 2 in 1 Type-C & Micro USB S... | Computers&Accessories\|Accessories&Peripherals\|... | ₹329 |
| 4 | B08CF3B7N1 | Portronics Konnect L 1.2M Fast Charging 3A 8 P... | Computers&Accessories\|Accessories&Peripherals\|... | ₹154 |

| | actual_price | discount_percentage | rating | rating_count | about_product |
|---|---|---|---|---|---|
| 0 | ₹1,099 | 64% | 4.2 | 24,269 | High Compatibility : Compatible With iPhone 12... |
| 1 | ₹349 | 43% | 4.0 | 43,994 | Compatible with all Type C enabled devices, be... |
| 2 | ₹1,899 | 90% | 3.9 | 7,928 | 【 Fast Charger& Data Sync】-With built-in safet... |
| 3 | ₹699 | 53% | 4.2 | 94,363 | The boAt Deuce USB 300 2 in 1 cable is compati... |
| 4 | ₹399 | 61% | 4.2 | 16,905 | [CHARGE & SYNC FUNCTION]- This cable comes wit... |

| | user_id | user_name | review_id | review_title |
|---|---|---|---|---|
| 0 | AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB... | Manav,Adarsh gupta,Sundeep,S.Sayeed Ahmed,jasp... | R3HXWT0LRP0NMF,R2AJM3LFTLZHFO,R6AQJGUP6P86,R1K... | Satisfied,Charging is really fast,Value for mo... |
| 1 | AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGYYVPDD7YG7FYNBX... | ArdKn,Nirbhay kumar,Sagar Viswanathan,Asp,Plac... | RGIQEG07R9HS2,R1SMWZQ86XIN8U,R2J3Y1WL29GWDE,RY... | A Good Braided Cable for Your Type C Device,Go... |
| 2 | AGU3BBQ2V2DDAMOAKGFAWDDQ6QHA,AESFLDV2PT363T2AQ... | Kunal,Himanshu,viswanath,sai niharka,saqib mal... | R3J3EQQ9TZI5ZJ,R3E7WBGK7ID0KV,RWU79XKQ6I1QF,R2... | Good speed for earlier versions,Good Product,W... |
| 3 | AEWAZDZZJLQUYVOVGBEUKSLXHQ5A,AG5HTSFRRE6NL3M5S... | Omkar dhale,JD,HEMALATHA,Ajwadh a.,amar singh ... | R3EEUZKKK9J36I,R3HJVYCLYOY554,REDECAZ7AMPQC,R1... | Good product,Good one,Nice,Really nice product... |
| 4 | AE3Q6KSUK5P75D5HFYHCRAOLODSA,AFUGIFH5ZAFXRDSZH... | rahuls6099,Swasat Borah,Ajay Wadke,Pranali,RVK... | R1BP4L2HH9TFUP,R16PVJEXKV6QZS,R2UPDB81N66T4P,R... | As good as original,Decent,Good one for second... |

1. **Python Notebook** - AWS connection successful

```
        'amazon_cleaned.csv'
    )
    print(f"✅ Uploaded: amazon_cleaned.csv to s3://{bucket_name}/")

    # Upload sample as well
    s3.upload_file(
        os.path.join(base_path, 'data', 'amazon_sample.csv'),
        bucket_name,
        'amazon_sample.csv'
    )
    print(f"✅ Uploaded: amazon_sample.csv to s3://{bucket_name}/")

    # Verify upload
    response = s3.list_objects_v2(Bucket=bucket_name)
    print(f"\n📂 Files in bucket:")
    for obj in response.get('Contents', []):
        size_kb = obj['Size'] / 1024
        print(f"   📄 {obj['Key']} - {size_kb:.1f} KB")

except Exception as e:
    print(f"❌ Upload failed: {e}")
```

```
✅ Uploaded: amazon_cleaned.csv to s3://amazon-profit-6663/
✅ Uploaded: amazon_sample.csv to s3://amazon-profit-6663/

📂 Files in bucket:
   📄 amazon_cleaned.csv - 4699.5 KB
   📄 amazon_sample.csv - 327.3 KB
```

In [9]:

```
# ============================================
# CELL 6: Create Athena table (FIXED)
# ============================================

import time

# Athena client with region specified
athena = boto3.client('athena', region_name='ap-south-1')

# Get bucket name
with open('bucket_name.txt', 'r') as f:
    bucket_name = f.read().strip()

# Create database
create_db_query = "CREATE DATABASE IF NOT EXISTS amazon_analysis"

try:
    response = athena.start_query_execution(
        QueryString=create_db_query,
        ResultConfiguration={'OutputLocation': f's3://{bucket_name}/athena-resul
    )
    print("✅ Database created/verified")
    time.sleep(2)
except Exception as e:
    print(f"❌ Error creating database: {e}")

# Create table
create_table_query = f"""
CREATE EXTERNAL TABLE IF NOT EXISTS amazon_analysis.products (
    product_id STRING,
    actual_price DOUBLE,
```

```
except:
    print("❌ AWS not configured")
```

```
✅ Setup complete
✅ Loaded 1,465 products
Columns: ['product_id', 'product_name', 'category', 'discounted_price', 'actual_price',
'discount_percentage', 'rating', 'rating_count', 'about_product', 'user_id', 'user_name',
'review_id', 'review_title', 'review_content', 'img_link', 'product_link',
'original_price', 'estimated_cost', 'profit_per_unit', 'margin_percent',
'estimated_revenue', 'estimated_profit']
✅ AWS configured
```

In [14]:
```
import boto3

sts = boto3.client('sts')
identity = sts.get_caller_identity()
print(f"✅ Connected! Account: {identity['Account']}")
```

```
✅ Connected! Account: 218504945626
```

In [15]:
```
# Create S3 bucket


import boto3
import random

s3 = boto3.client('s3')


bucket_name = f"amazon-profit-{random.randint(1000,9999)}"

try:
    # Create bucket (for ap-south-1 region)
    s3.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={'LocationConstraint': 'ap-south-1'}
    )
    print(f"✅ Bucket created: {bucket_name}")

    # Save bucket name for later use
    with open('bucket_name.txt', 'w') as f:
        f.write(bucket_name)

except Exception as e:
    print(f"❌ Error creating bucket: {e}")
```

```
✅ Bucket created: amazon-profit-6663
```

In [16]:
```
# : Upload cleaned data to S3

import os

# Get bucket name
with open('bucket_name.txt', 'r') as f:
    bucket_name = f.read().strip()

# Upload cleaned data
try:
    s3.upload_file(
        os.path.join(base_path, 'data', 'amazon_cleaned.csv'),
        bucket_name,
```

```
    identity = sts.get_caller_identity()
    print(f"✅ AWS Connected Successfully!")
    print(f"   Account: {identity['Account']}")
    print(f"   User ARN: {identity['Arn']}")

    # Check region
    session = boto3.session.Session()
    region = session.region_name
    print(f"   Region: {region}")

except Exception as e:
    print("❌ AWS Not Configured!")
    print("\n🔧 Run this in terminal:")
    print("   aws configure")
    print("\n   Enter your AWS Access Key ID")
    print("   Enter your AWS Secret Access Key")
    print("   Default region: ap-south-1")
    print("   Output format: json")
```

```
🔍 Checking AWS Configuration...
==================================================
✅ AWS Connected Successfully!
   Account: 218504945626
   User ARN: arn:aws:iam::218504945626:user/data_analyst
   Region: ap-south-1
```

In [12]:
```
# Day 2 notebook mein yeh line yaad rakhna:
df = pd.read_csv(r"C:\Users\ACER\Desktop\Amazon_Project\data\amazon_cleaned.csv"
```

In [13]:
```
# CELL 1: Imports and Setup
import pandas as pd
import numpy as np
import boto3
import os
import time
import random
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime

# Paths
base_path = r"C:\Users\ACER\Desktop\Amazon_Project"
data_path = os.path.join(base_path, 'data', 'amazon_cleaned.csv')
screenshots_path = os.path.join(base_path, 'screenshots')

# Create screenshots folder if not exists
os.makedirs(screenshots_path, exist_ok=True)

print("✅ Setup complete")

# CELL 2: Load Cleaned Data
df = pd.read_csv(data_path)
print(f"✅ Loaded {len(df):,} products")
print(f"Columns: {df.columns.tolist()}")

# CELL 3: AWS Check
try:
    s3 = boto3.client('s3')
    print("✅ AWS configured")
```

```
✅ Project path: C:\Users\ACER\Desktop\Amazon_Project
✅ Data file: C:\Users\ACER\Desktop\Amazon_Project\data\amazon_cleaned.csv
✅ Screenshots: C:\Users\ACER\Desktop\Amazon_Project\screenshots

✅ Loaded cleaned data: 1,465 rows
   product_id                                       product_name  \
0  B07JW9H4J1  Wayona Nylon Braided USB to Lightning Fast Cha...
1  B098NS6PVG  Ambrane Unbreakable 60W / 3A Fast Charging 1.5...

                                            category discounted_price  \
0  Computers&Accessories|Accessories&Peripherals|...             ₹399
1  Computers&Accessories|Accessories&Peripherals|...             ₹199

   actual_price  discount_percentage  rating  rating_count  \
0        1099.0                   64     4.2       24269.0
1         349.0                   43     4.0       43994.0

                                      about_product  \
0  High Compatibility : Compatible With iPhone 12...
1  Compatible with all Type C enabled devices, be...

                                         user_id  ...  \
0  AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBB...  ...
1  AECPFYFQVRUWC3KGNLJIOREFP5LQ,AGYYVPDD7YG7FYNBX...  ...

                                       review_title  \
0  Satisfied,Charging is really fast,Value for mo...
1  A Good Braided Cable for Your Type C Device,Go...

                                     review_content  \
0  Looks durable Charging is fine tooNo complains...
1  I ordered this cable to connect my phone to An...

                                         img_link  \
0  https://m.media-amazon.com/images/W/WEBP_40237...
1  https://m.media-amazon.com/images/W/WEBP_40237...

                                       product_link original_price  \
0  https://www.amazon.in/Wayona-Braided-WN3LG1-Sy...        3052.78
1  https://www.amazon.in/Ambrane-Unbreakable-Char...         612.28

  estimated_cost  profit_per_unit  margin_percent  estimated_revenue  \
0        1221.11          -122.11          -11.11         26671631.0
1         244.91           104.09           29.83         15353906.0

   estimated_profit
0      -2963487.59
1       4579335.46

[2 rows x 22 columns]
```

In [11]:

```
#  Check AWS configuration


print("🔍 Checking AWS Configuration...")
print("="*50)

try:
    # Test AWS connection
    sts = boto3.client('sts')
```

In [10]:

```
import pandas as pd
import numpy as np
import boto3
import os
import time
import random
from datetime import datetime

# Project paths
base_path = r"C:\Users\ACER\Desktop\Amazon_Project"
data_path = os.path.join(base_path, 'data', 'amazon_cleaned.csv')
screenshots_path = os.path.join(base_path, 'screenshots')

print(f"✅ Project path: {base_path}")
print(f"✅ Data file: {data_path}")
print(f"✅ Screenshots: {screenshots_path}")

# Load cleaned data from Day 1
df = pd.read_csv(data_path)
print(f"\n✅ Loaded cleaned data: {len(df):,} rows")
print(df.head(2))
```

```
    discount_percentage DOUBLE,
    rating DOUBLE,
    rating_count INT,
    original_price DOUBLE,
    estimated_cost DOUBLE,
    profit_per_unit DOUBLE,
    margin_percent DOUBLE,
    estimated_revenue DOUBLE,
    estimated_profit DOUBLE
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES ('field.delim' = ',')
LOCATION 's3://{bucket_name}/'
TBLPROPERTIES ('skip.header.line.count' = '1')
"""

try:
    response = athena.start_query_execution(
        QueryString=create_table_query,
        QueryExecutionContext={'Database': 'amazon_analysis'},
        ResultConfiguration={'OutputLocation': f's3://{bucket_name}/athena-resul
    )
    print("✅ Table created successfully!")

    # Get query execution ID
    query_id = response['QueryExecutionId']
    print(f"   Query ID: {query_id}")

except Exception as e:
    print(f"❌ Error creating table: {e}")
```

```
✅ Database created/verified
✅ Table created successfully!
   Query ID: 42cb07f9-a092-4558-a936-d7fb34cfb805
```

In [ ]: