DavidJetter_Product_Analysis

July 18, 2025

[ ]:
```
# Install all required packages for the product performance analysis
!pip install --upgrade pip

# Core data processing and analysis libraries
!pip install pandas numpy

# Google Cloud and BigQuery libraries
!pip install google-cloud-bigquery
!pip install bigframes
!pip install db-dtypes

# Visualization libraries
!pip install matplotlib seaborn

# Interactive widgets for Jupyter
!pip install ipywidgets

# Additional utilities
!pip install pyarrow
!pip install google-auth
!pip install google-auth-oauthlib
!pip install google-auth-httplib2

print(" All packages installed successfully!")
```

[4]:
```
import bigframes.pandas as bf
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from datetime import datetime
import warnings
from IPython.display import display, clear_output
import ipywidgets as widgets
from ipywidgets import Button, VBox, HBox, HTML
warnings.filterwarnings('ignore')

# ================================================================================
```

```
# SETUP AND AUTHENTICATION
# ================================================================================

# Set BigQuery project BEFORE any data loading
PROJECT_ID = "silken-apex-465710-e4"  # Your project ID
bf.options.bigquery.project = PROJECT_ID

print(" Configuring BigQuery connection...")
print(f" Project ID: {PROJECT_ID}")

# Set matplotlib style to ggplot2
plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['font.size'] = 11
plt.rcParams['axes.titlesize'] = 14
plt.rcParams['axes.labelsize'] = 12

# ================================================================================
# DATA LOADING WITH ERROR HANDLING
# ================================================================================

def load_superstore_data():
    """Load Superstore data with comprehensive error handling"""

    print("Loading Superstore data...")

    try:
        # Primary attempt: Load from BigQuery
        table_id = f"{PROJECT_ID}.SuperstoreData.Superstore"
        superstore_df = bf.read_gbq(table_id)
        superstore_pd = superstore_df.to_pandas()

        print(" BigQuery data loaded successfully!")
        print(f" Dataset shape: {superstore_pd.shape}")
        return superstore_pd

    except Exception as e:
        print(f" BigQuery loading failed: {e}")
        print("\n Using sample data for demonstration...")

        # Fallback: Create sample data for demonstration
        return create_sample_data()

def create_sample_data():
    """Create sample Superstore data for demonstration"""

    np.random.seed(42)
```

```
    n_records = 1000

    categories = ['Technology', 'Office Supplies', 'Furniture']
    segments = ['Consumer', 'Corporate', 'Home Office']
    regions = ['Central', 'East', 'South', 'West']

    sample_data = {
        'order_id': [f'US-2023-{i:06d}' for i in range(n_records)],
        'order_date': pd.date_range('2023-01-01', periods=n_records, freq='D'),
        'ship_date': pd.date_range('2023-01-03', periods=n_records, freq='D'),
        'customer': [f'Customer-{i}' for i in range(n_records)],
        'manufactory': np.random.choice(['HP', 'Canon', 'Logitech', 'Other'],␣
 ↪n_records),
        'product_name': [f'Product-{i}' for i in range(n_records)],
        'segment': np.random.choice(segments, n_records),
        'category': np.random.choice(categories, n_records),
        'subcategory': np.random.choice(['Accessories', 'Paper', 'Storage',␣
 ↪'Phones'], n_records),
        'region': np.random.choice(regions, n_records),
        'zip': np.random.randint(10000, 99999, n_records),
        'city': [f'City-{i}' for i in range(n_records)],
        'state': np.random.choice(['CA', 'NY', 'TX', 'FL'], n_records),
        'country': ['United States'] * n_records,
        'discount': np.random.beta(2, 8, n_records) * 0.5,
        'quantity': np.random.randint(1, 10, n_records),
        'sales': np.random.lognormal(4, 1, n_records),
        'profit': np.random.normal(50, 100, n_records)
    }

    df = pd.DataFrame(sample_data)
    df['profit_margin'] = df['profit'] / df['sales']

    print(" Sample data created for demonstration")
    print(f" Sample dataset shape: {df.shape}")

    return df

# Load data (with fallback to sample data if BigQuery fails)
superstore_pd = load_superstore_data()

# Simple data preparation
superstore_pd['order_date'] = pd.to_datetime(superstore_pd['order_date'])
superstore_pd = superstore_pd.fillna(0)

# ================================================================================
# INTERACTIVE NAVIGATION SYSTEM
# ================================================================================
```

```python
# Global variables for navigation
questions = [
    "Which products/categories perform best?",
    "What drives product profitability?",
    "Where should we focus product strategy?"
]

def create_navigation_buttons():
    """Create interactive navigation buttons"""
    next_button = Button(
        description='Next Question →',
        disabled=False,
        button_style='success',
        tooltip='Click to proceed to next analysis',
        layout=widgets.Layout(width='200px', height='40px')
    )

    prev_button = Button(
        description='← Previous Question',
        disabled=True,
        button_style='info',
        tooltip='Click to go back to previous analysis',
        layout=widgets.Layout(width='200px', height='40px')
    )

    return prev_button, next_button

def display_question_header(question_num, question_text):
    """Display styled header for each question"""
    header_html = f"""
    <div style="background: linear-gradient(90deg, #2E86AB, #A23B72);
                padding: 20px;
                border-radius: 10px;
                margin: 20px 0;">
        <h2 style="color: white; margin: 0; text-align: center;">
            Question {question_num + 1}: {question_text}
        </h2>
    </div>
    """
    display(HTML(header_html))

# ================================================================================
# QUESTION 1: Which products/categories perform best?
# ================================================================================

def analyze_question_1():
```

```
    """First analytical question with full dashboard"""
    clear_output(wait=True)
    display_question_header(0, questions[0])

    print("  ANALYZING: Product & Category Performance")
    print("="*60)

    # Category Performance Analysis
    category_performance = superstore_pd.groupby('category').agg({
        'sales': ['sum', 'mean', 'count'],
        'profit': ['sum', 'mean'],
        'profit_margin': 'mean'
    }).round(4)

    category_performance.columns = ['Total_Sales', 'Avg_Order_Value',␣
↪'Order_Count',
                                    'Total_Profit', 'Avg_Profit_Per_Order',␣
↪'Avg_Profit_Margin']
    category_performance = category_performance.sort_values('Total_Profit',␣
↪ascending=False)

    # Create comprehensive dashboard
    fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(16, 12))
    fig.suptitle('Product & Category Performance Dashboard', fontsize=16,␣
↪fontweight='bold')

    # Chart 1: Category Sales vs Profit (Dual Axis)
    categories = category_performance.index
    x_pos = np.arange(len(categories))

    ax1_twin = ax1.twinx()
    bars1 = ax1.bar(x_pos - 0.2, category_performance['Total_Sales'], 0.4,
                    label='Total Sales', alpha=0.8, color='#2E86AB')
    bars2 = ax1_twin.bar(x_pos + 0.2, category_performance['Total_Profit'], 0.4,
                         label='Total Profit', alpha=0.8, color='#A23B72')

    ax1.set_xlabel('Category')
    ax1.set_ylabel('Total Sales ($)', color='#2E86AB')
    ax1_twin.set_ylabel('Total Profit ($)', color='#A23B72')
    ax1.set_title('Sales vs Profit by Category')
    ax1.set_xticks(x_pos)
    ax1.set_xticklabels(categories, rotation=45)
    ax1.legend(loc='upper left')
    ax1_twin.legend(loc='upper right')

    # Chart 2: Profit Margin by Category
    colors = ['#2E86AB', '#A23B72', '#F18F01'][:len(categories)]
```

```
    bars = ax2.bar(categories, category_performance['Avg_Profit_Margin'],
                   color=colors, alpha=0.8)
    ax2.set_title('Average Profit Margin by Category')
    ax2.set_ylabel('Profit Margin')
    ax2.set_xticklabels(categories, rotation=45)

    # Add value labels on bars
    for bar in bars:
        height = bar.get_height()
        ax2.text(bar.get_x() + bar.get_width()/2., height,
                f'{height:.1%}', ha='center', va='bottom')

    # Chart 3: Top 10 Products by Profit
    product_performance = superstore_pd.groupby('product_name').agg({
        'profit': 'sum',
        'sales': 'sum',
        'order_id': 'count'
    }).sort_values('profit', ascending=False).head(10)

    product_names = [name[:30] + '...' if len(name) > 30 else name
                     for name in product_performance.index]

    ax3.barh(range(len(product_names)), product_performance['profit'],
             color='#F18F01', alpha=0.8)
    ax3.set_yticks(range(len(product_names)))
    ax3.set_yticklabels(product_names, fontsize=9)
    ax3.set_xlabel('Total Profit ($)')
    ax3.set_title('Top 10 Products by Total Profit')
    ax3.invert_yaxis()

    # Chart 4: Category Order Volume Distribution
    ax4.pie(category_performance['Order_Count'], labels=categories, autopct='%1.
↪1f%%',
            colors=colors, startangle=90)
    ax4.set_title('Order Distribution by Category')

    plt.tight_layout()
    plt.show()

    # Detailed Analytical Insights
    print("\n KEY FINDINGS:")
    print("-" * 40)

    best_category = category_performance.index[0]
    best_profit = category_performance.loc[best_category, 'Total_Profit']
    best_margin = category_performance.loc[best_category, 'Avg_Profit_Margin']
```

```
    print(f"  TOP PERFORMING CATEGORY: {best_category}")
    print(f"    • Total Profit: ${best_profit:,.2f}")
    print(f"    • Profit Margin: {best_margin:.1%}")
    print(f"    • Market Share: {category_performance.loc[best_category,␣
↪'Order_Count']/category_performance['Order_Count'].sum():.1%}")

    print(f"\n  COMPLETE CATEGORY RANKINGS:")
    for i, (category, row) in enumerate(category_performance.iterrows(), 1):
        print(f"    {i}. {category}")
        print(f"       → Profit: ${row['Total_Profit']:,.0f}␣
↪({row['Avg_Profit_Margin']:.1%} margin)")
        print(f"       → Sales: ${row['Total_Sales']:,.0f} ({row['Order_Count']:
↪,} orders)")

    if len(product_performance) > 0:
        top_product = product_performance.index[0]
        top_product_profit = product_performance.loc[top_product, 'profit']
        print(f"\n  TOP INDIVIDUAL PRODUCT:")
        print(f"    • Product: {top_product[:50]}...")
        print(f"    • Total Profit: ${top_product_profit:,.2f}")
        print(f"    • Orders: {product_performance.loc[top_product, 'order_id']:
↪,}")

    # Interactive Navigation Buttons
    prev_btn, next_btn = create_navigation_buttons()
    prev_btn.disabled = True

    def on_next_click(b):
        analyze_question_2()

    next_btn.on_click(on_next_click)

    # Display navigation
    button_box = HBox([prev_btn, next_btn], layout=widgets.
↪Layout(justify_content='center', margin='20px'))
    display(button_box)

# ================================================================================
# QUESTION 2: What drives product profitability?
# ================================================================================

def analyze_question_2():
    """Second analytical question with profitability drivers"""
    clear_output(wait=True)
    display_question_header(1, questions[1])

    print("  ANALYZING: Product Profitability Drivers")
```

```
    print("="*60)

    # Aggressive data cleaning for reliable analysis
    clean_data = superstore_pd[
        (superstore_pd['discount'].notna()) &
        (superstore_pd['profit_margin'].notna()) &
        (superstore_pd['quantity'].notna()) &
        (superstore_pd['discount'] >= 0) &
        (superstore_pd['profit_margin'].between(-1, 2))
    ].copy()

    print(f"Using {len(clean_data)} clean records (removed {len(superstore_pd)- len(clean_data)} problematic rows)")

    # Create comprehensive profitability analysis dashboard
    fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(16, 12))
    fig.suptitle('Product Profitability Drivers Analysis', fontsize=16,fontweight='bold')

    # Chart 1: Discount Impact Analysis
    discount_ranges = [
        (0, 0.1, "0-10%"),
        (0.1, 0.2, "10-20%"),
        (0.2, 0.3, "20-30%"),
        (0.3, 0.5, "30-50%"),
        (0.5, 1.0, "50%+")
    ]

    avg_margins = []
    range_labels = []
    order_counts = []

    for min_val, max_val, label in discount_ranges:
        mask = (clean_data['discount'] >= min_val) & (clean_data['discount'] <max_val)
        subset = clean_data[mask]

        if len(subset) > 0:
            avg_margin = subset['profit_margin'].mean()
            count = len(subset)

            if pd.notna(avg_margin):
                avg_margins.append(avg_margin)
                range_labels.append(f"{label}\n({count} orders)")
                order_counts.append(count)

    bars = ax1.bar(range(len(range_labels)), avg_margins,
```

```
                color=['#E63946', '#F18F01', '#F77F00', '#FCBF49', '#D62828'],
                alpha=0.8)

    ax1.set_xlabel('Discount Range')
    ax1.set_ylabel('Average Profit Margin')
    ax1.set_title('Discount Impact on Profitability')
    ax1.set_xticks(range(len(range_labels)))
    ax1.set_xticklabels(range_labels)

    # Add value labels on bars
    for bar, margin in zip(bars, avg_margins):
        height = bar.get_height()
        ax1.text(bar.get_x() + bar.get_width()/2., height,
                f'{margin:.1%}', ha='center', va='bottom', fontweight='bold')

    # Chart 2: Category Profitability Comparison
    category_margins = []
    category_labels = []
    for category in clean_data['category'].unique():
        cat_data = clean_data[clean_data['category'] ==␣
↪category]['profit_margin']
        category_margins.append(cat_data.values)
        category_labels.append(category)

    box_plot = ax2.boxplot(category_margins, labels=category_labels,␣
↪patch_artist=True)
    colors = ['#2E86AB', '#A23B72', '#F18F01']
    for patch, color in zip(box_plot['boxes'], colors[:len(box_plot['boxes'])]):
        patch.set_facecolor(color)
        patch.set_alpha(0.7)

    ax2.set_title('Profit Margin Distribution by Category')
    ax2.set_ylabel('Profit Margin')
    ax2.tick_params(axis='x', rotation=45)

    # Chart 3: Quantity vs Profitability
    quantity_bins = pd.cut(clean_data['quantity'], bins=5)
    quantity_impact = clean_data.groupby(quantity_bins).agg({
        'profit_margin': 'mean',
        'order_id': 'count'
    })

    qty_labels = [f"{int(interval.left)}-{int(interval.right)}" for interval in␣
↪quantity_impact.index]

    ax3_twin = ax3.twinx()
    bars1 = ax3.bar(range(len(qty_labels)), quantity_impact['profit_margin'],
```

```
                  0.6, label='Avg Profit Margin', alpha=0.8, color='#2A9D8F')
    line1 = ax3_twin.plot(range(len(qty_labels)), quantity_impact['order_id'],
                          'ro-', label='Order Count', color='#F4A261',␣
↪linewidth=2)

    ax3.set_xlabel('Quantity Range')
    ax3.set_ylabel('Profit Margin', color='#2A9D8F')
    ax3_twin.set_ylabel('Order Count', color='#F4A261')
    ax3.set_title('Quantity Impact on Profitability')
    ax3.set_xticks(range(len(qty_labels)))
    ax3.set_xticklabels(qty_labels, rotation=45)
    ax3.legend(loc='upper left')
    ax3_twin.legend(loc='upper right')

    # Chart 4: Sales Volume vs Margin Trend
    sales_bins = pd.cut(clean_data['sales'], bins=10)
    sales_impact = clean_data.groupby(sales_bins).agg({
        'profit_margin': 'mean'
    })

    sales_midpoints = [(interval.left + interval.right) / 2 for interval in␣
↪sales_impact.index]

    ax4.plot(sales_midpoints, sales_impact['profit_margin'], marker='o',␣
↪linewidth=2,
             markersize=6, color='#457B9D')
    ax4.set_xlabel('Sales Volume ($)')
    ax4.set_ylabel('Average Profit Margin')
    ax4.set_title('Sales Volume vs Profit Margin Trend')
    ax4.grid(True, alpha=0.3)

    plt.tight_layout()
    plt.show()

    # Comprehensive Analytical Insights
    print("\n DETAILED PROFITABILITY INSIGHTS:")
    print("-" * 50)

    # Correlation analysis
    corr_discount = clean_data['discount'].corr(clean_data['profit_margin'])
    corr_quantity = clean_data['quantity'].corr(clean_data['profit_margin'])
    corr_sales = clean_data['sales'].corr(clean_data['profit_margin'])

    print(" CORRELATION ANALYSIS:")
    print(f"   • Discount vs Profit Margin: {corr_discount:.3f}")
    print(f"   • Quantity vs Profit Margin: {corr_quantity:.3f}")
    print(f"   • Sales Volume vs Profit Margin: {corr_sales:.3f}")
```

```
    # Optimal strategy identification
    if avg_margins:
        best_range_idx = avg_margins.index(max(avg_margins))
        best_range = range_labels[best_range_idx]
        best_margin = max(avg_margins)

        print(f"\n OPTIMAL DISCOUNT STRATEGY:")
        print(f"    • Best Discount Range: {best_range.split('(')[0].strip()}")
        print(f"    • Achieves: {best_margin:.1%} average profit margin")
        print(f"    • Order Volume: {order_counts[best_range_idx]:,} orders")

    # Category-specific insights
    print(f"\n CATEGORY-SPECIFIC INSIGHTS:")
    for category in clean_data['category'].unique():
        cat_data = clean_data[clean_data['category'] == category]
        avg_margin = cat_data['profit_margin'].mean()
        avg_discount = cat_data['discount'].mean()
        avg_quantity = cat_data['quantity'].mean()
        print(f"    • {category}:")
        print(f"      → Avg Margin: {avg_margin:.1%}")
        print(f"      → Avg Discount: {avg_discount:.1%}")
        print(f"      → Avg Quantity: {avg_quantity:.1f}")

    # Interactive Navigation Buttons
    prev_btn, next_btn = create_navigation_buttons()

    def on_prev_click(b):
        analyze_question_1()

    def on_next_click(b):
        analyze_question_3()

    prev_btn.on_click(on_prev_click)
    next_btn.on_click(on_next_click)

    # Display navigation
    button_box = HBox([prev_btn, next_btn], layout=widgets.
 ↪Layout(justify_content='center', margin='20px'))
    display(button_box)

# ================================================================================
# QUESTION 3: Where should we focus product strategy?
# ================================================================================

def analyze_question_3():
    """Third analytical question with strategic recommendations"""
```

```
    clear_output(wait=True)
    display_question_header(2, questions[2])

    print("  STRATEGIC FOCUS ANALYSIS")
    print("="*60)

    # Strategic portfolio analysis
    fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(16, 12))
    fig.suptitle('Strategic Product Portfolio Analysis', fontsize=16,␣
↪fontweight='bold')

    # Calculate product metrics for BCG-style analysis
    product_metrics = superstore_pd.groupby('product_name').agg({
        'profit': 'sum',
        'sales': 'sum',
        'order_id': 'count',
        'profit_margin': 'mean'
    })

    # Define thresholds for portfolio categorization
    profit_threshold = product_metrics['profit'].median()
    volume_threshold = product_metrics['order_id'].median()

    # Categorize products into BCG matrix
    stars = product_metrics[(product_metrics['profit'] > profit_threshold) &
                            (product_metrics['order_id'] > volume_threshold)]
    cash_cows = product_metrics[(product_metrics['profit'] > profit_threshold) &
                                (product_metrics['order_id'] <=␣
↪volume_threshold)]
    question_marks = product_metrics[(product_metrics['profit'] <=␣
↪profit_threshold) &
                                     (product_metrics['order_id'] >␣
↪volume_threshold)]
    dogs = product_metrics[(product_metrics['profit'] <= profit_threshold) &
                           (product_metrics['order_id'] <= volume_threshold)]

    # Chart 1: BCG Portfolio Matrix
    ax1.scatter(stars['order_id'], stars['profit'],
                s=100, alpha=0.7, c='#2E86AB', label='Stars', edgecolors='black')
    ax1.scatter(cash_cows['order_id'], cash_cows['profit'],
                s=100, alpha=0.7, c='#A23B72', label='Cash Cows',␣
↪edgecolors='black')
    ax1.scatter(question_marks['order_id'], question_marks['profit'],
                s=100, alpha=0.7, c='#F18F01', label='Question Marks',␣
↪edgecolors='black')
    ax1.scatter(dogs['order_id'], dogs['profit'],
```

```
                s=100, alpha=0.7, c='#E63946', label='Dogs', edgecolors='black')

    ax1.axhline(y=profit_threshold, color='gray', linestyle='--', alpha=0.7)
    ax1.axvline(x=volume_threshold, color='gray', linestyle='--', alpha=0.7)

    ax1.set_xlabel('Order Volume (Count)')
    ax1.set_ylabel('Total Profit ($)')
    ax1.set_title('BCG Product Portfolio Matrix')
    ax1.legend()
    ax1.grid(True, alpha=0.3)

    # Chart 2: Portfolio Distribution
    portfolio_counts = [len(stars), len(cash_cows), len(question_marks),␣
↪len(dogs)]
    portfolio_labels = ['Stars', 'Cash Cows', 'Question Marks', 'Dogs']
    colors = ['#2E86AB', '#A23B72', '#F18F01', '#E63946']

    wedges, texts, autotexts = ax2.pie(portfolio_counts,␣
↪labels=portfolio_labels,
                                       autopct='%1.1f%%', colors=colors,␣
↪startangle=90)
    ax2.set_title('Product Portfolio Distribution')

    # Chart 3: Revenue Contribution by Portfolio
    revenue_contribution = [stars['sales'].sum(), cash_cows['sales'].sum(),
                            question_marks['sales'].sum(), dogs['sales'].sum()]

    bars = ax3.bar(portfolio_labels, revenue_contribution, color=colors,␣
↪alpha=0.8)
    ax3.set_title('Revenue Contribution by Portfolio Segment')
    ax3.set_ylabel('Total Sales ($)')
    ax3.tick_params(axis='x', rotation=45)

    # Add value labels on bars
    for bar in bars:
        height = bar.get_height()
        ax3.text(bar.get_x() + bar.get_width()/2., height,
                f'${height/1000:.0f}K', ha='center', va='bottom')

    # Chart 4: Category Strategic Priority Matrix
    category_priority = superstore_pd.groupby('category').agg({
        'profit': 'sum',
        'profit_margin': 'mean',
        'sales': 'sum'
    }).sort_values('profit', ascending=False)

    priority_scores = []
```

```
    for cat in category_priority.index:
        profit_score = category_priority.loc[cat, 'profit'] /␣
↪category_priority['profit'].max()
        margin_score = category_priority.loc[cat, 'profit_margin'] /␣
↪category_priority['profit_margin'].max()
        priority_score = (profit_score * 0.6) + (margin_score * 0.4)
        priority_scores.append(priority_score)

    bars = ax4.barh(range(len(category_priority.index)), priority_scores,
                    color=['#2E86AB', '#A23B72', '#F18F01'][:
↪len(category_priority.index)])
    ax4.set_yticks(range(len(category_priority.index)))
    ax4.set_yticklabels(category_priority.index)
    ax4.set_xlabel('Strategic Priority Score')
    ax4.set_title('Category Investment Priority Ranking')
    ax4.set_xlim(0, 1)

    # Add score labels
    for i, (bar, score) in enumerate(zip(bars, priority_scores)):
        ax4.text(score + 0.02, bar.get_y() + bar.get_height()/2,
                 f'{score:.2f}', va='center')

    plt.tight_layout()
    plt.show()

    # Comprehensive Strategic Recommendations
    print("\n DETAILED STRATEGIC RECOMMENDATIONS:")
    print("-" * 60)

    print(f" STARS ({len(stars)} products) - INVEST & GROW:")
    if len(stars) > 0:
        top_star = stars['profit'].idxmax()
        star_profit = stars.loc[top_star, 'profit']
        star_volume = stars.loc[top_star, 'order_id']
        print(f"   • Top Star Product: {top_star[:50]}...")
        print(f"   • Profit Contribution: ${star_profit:,.2f}")
        print(f"   • Order Volume: {star_volume:,}")
        print(f"   • Strategy: Increase marketing spend by 25-30%")
        print(f"   • Investment Priority: HIGH")
    else:
        print(f"   • No star products identified")
        print(f"   • Strategy: Develop high-potential products")

    print(f"\n CASH COWS ({len(cash_cows)} products) - HARVEST:")
    if len(cash_cows) > 0:
        top_cow = cash_cows['profit'].idxmax()
        cow_profit = cash_cows.loc[top_cow, 'profit']
```

```
        cow_volume = cash_cows.loc[top_cow, 'order_id']
        print(f"   • Top Cash Cow: {top_cow[:50]}...")
        print(f"   • Profit Contribution: ${cow_profit:,.2f}")
        print(f"   • Order Volume: {cow_volume:,}")
        print(f"   • Strategy: Optimize operations, maintain market position")
        print(f"   • Investment Priority: MEDIUM")
    else:
        print(f"   • No cash cow products identified")

    print(f"\n QUESTION MARKS ({len(question_marks)} products) - SELECTIVE␣
↪INVESTMENT:")
    if len(question_marks) > 0:
        qm_total_volume = question_marks['order_id'].sum()
        qm_avg_margin = question_marks['profit_margin'].mean()
        print(f"   • Total Volume: {qm_total_volume:,} orders")
        print(f"   • Average Margin: {qm_avg_margin:.1%}")
        print(f"   • Strategy: Test price optimization, improve profit margins")
        print(f"   • Investment Priority: SELECTIVE")
    else:
        print(f"   • No question mark products identified")

    print(f"\n DOGS ({len(dogs)} products) - DIVEST OR REPOSITION:")
    if len(dogs) > 0:
        dog_total_loss = dogs[dogs['profit'] < 0]['profit'].sum()
        print(f"   • Products to Review: {len(dogs):,}")
        print(f"   • Total Loss from Unprofitable: ${abs(dog_total_loss):,.2f}")
        print(f"   • Strategy: Discontinue worst performers, find niche markets␣
↪for others")
        print(f"   • Investment Priority: LOW/DIVEST")

    print(f"\n CATEGORY INVESTMENT PRIORITIES:")
    for i, (category, score) in enumerate(zip(category_priority.index,␣
↪priority_scores), 1):
        priority_level = "HIGH" if score > 0.7 else "MEDIUM" if score > 0.4␣
↪else "LOW"
        category_profit = category_priority.loc[category, 'profit']
        category_margin = category_priority.loc[category, 'profit_margin']
        print(f"   {i}. {category}:")
        print(f"      → Priority Score: {score:.2f} ({priority_level})")
        print(f"      → Total Profit: ${category_profit:,.2f}")
        print(f"      → Avg Margin: {category_margin:.1%}")

    print(f"\n IMMEDIATE ACTION PLAN (Next 30 Days):")
    print(f"   1. Focus investment on {category_priority.index[0]} category")
    print(f"   2. Launch pricing optimization pilot for Question Mark products")
    print(f"   3. Increase marketing budget for top 5 Star products by 25%")
    print(f"   4. Conduct profitability review for bottom 20% of Dog products")
```

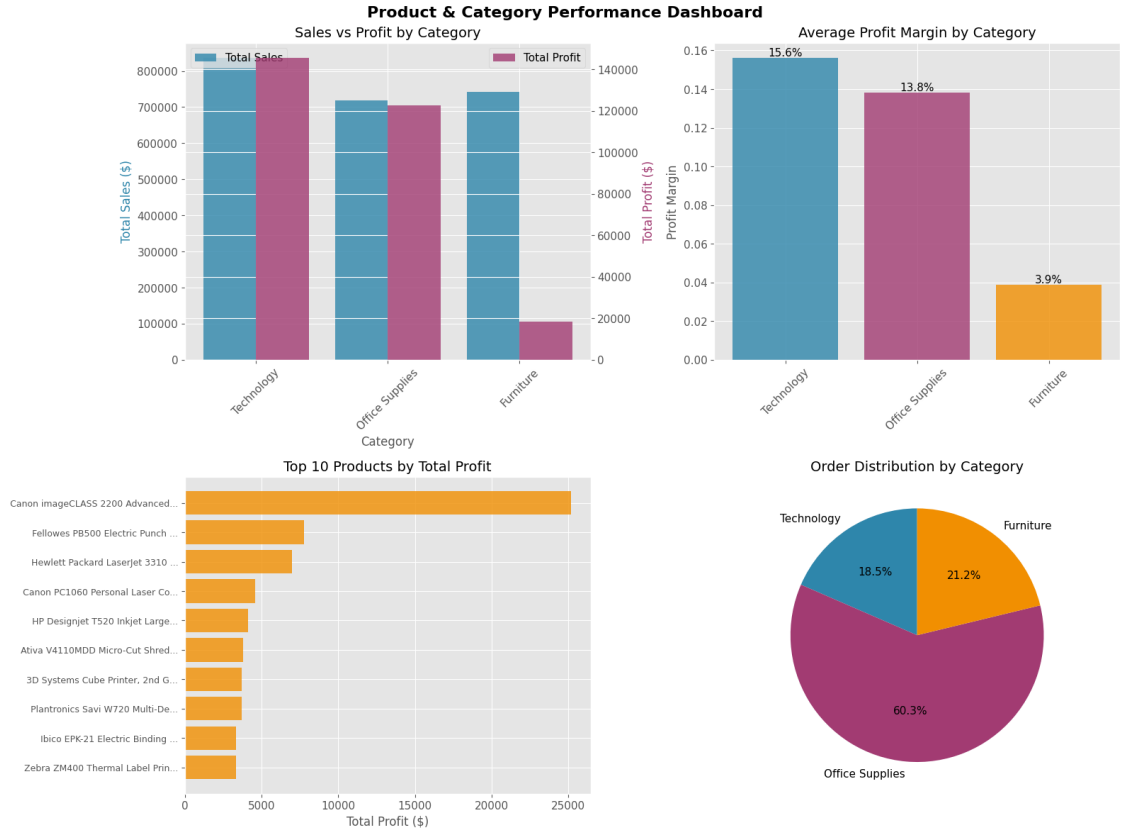
```
    print(f"   5. Optimize operations for top 3 Cash Cow products")

    print(f"\n STRATEGIC INITIATIVES (90-Day Roadmap):")
    print(f"   1. Launch premium product line in {category_priority.index[0]}␣
 ↪category")
    print(f"   2. Implement dynamic pricing algorithm for high-volume products")
    print(f"   3. Develop customer retention program targeting Cash Cow buyers")
    print(f"   4. Exit bottom 10% of unprofitable products")
    print(f"   5. Invest in R&D for next-generation Star products")

    # Final Navigation Buttons
    prev_btn, next_btn = create_navigation_buttons()
    prev_btn.disabled = False
    next_btn.disabled = True
    next_btn.description = 'Analysis Complete  '
    next_btn.button_style = 'success'

    def on_prev_click(b):
        analyze_question_2()

    prev_btn.on_click(on_prev_click)

    # Display final navigation
    button_box = HBox([prev_btn, next_btn], layout=widgets.
 ↪Layout(justify_content='center', margin='20px'))
    display(button_box)

# ================================================================================
# MAIN EXECUTION & STARTUP
# ================================================================================

def start_analysis():
    """Initialize and start the interactive analysis"""
    title_html = """
    <div style="background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
                padding: 30px;
                border-radius: 15px;
                text-align: center;
                margin: 20px 0;">
        <h1 style="color: white; margin: 0; font-size: 28px;">
             Interactive Product Performance Analysis
        </h1>
        <p style="color: white; margin: 10px 0; font-size: 16px;">
            Navigate through three core analytical questions with interactive␣
 ↪visualizations
        </p>
        <p style="color: #E8E8E8; margin: 0; font-size: 14px;">
```

```
            Click the navigation buttons to move between questions and explore␣
 ↪insights
        </p>
    </div>
    """

    overview_html = """
    <div style="background: #f8f9fa;
                padding: 20px;
                border-left: 5px solid #667eea;
                margin: 20px 0;">
        <h3 style="color: #495057; margin-top: 0;">  Interactive Analysis␣
 ↪Framework</h3>
        <ul style="color: #6c757d;">
            <li><strong>Question 1:</strong> Which products/categories perform␣
 ↪best?</li>
            <li><strong>Question 2:</strong> What drives product profitability?
 ↪</li>
            <li><strong>Question 3:</strong> Where should we focus product␣
 ↪strategy?</li>
        </ul>
        <p style="color: #6c757d; margin-bottom: 0;">
            <em>Use the Previous/Next buttons to navigate between questions and␣
 ↪dive deep into each analysis.</em>
        </p>
    </div>
    """

    display(HTML(title_html))
    display(HTML(overview_html))

    # Start with first question
    analyze_question_1()

# Initialize and start the complete interactive analysis
print("  Initializing Interactive Product Performance Analysis...")
print("  Data processing complete!")
print("  Using matplotlib with ggplot2 styling")
print("  Interactive navigation buttons enabled")
print("  Ready to begin comprehensive analysis...")
print("\n" + "="*60)

start_analysis()
```

```
HTML(value='\n    <div style="background: linear-gradient(90deg, #2E86AB,␣
 ↪#A23B72); \n                padding:…
```

```
 ANALYZING: Product & Category Performance
```

==========



KEY FINDINGS:

TOP PERFORMING CATEGORY: Technology

- Total Profit: $145,454.95
- Profit Margin: 15.6%
- Market Share: 18.5%

COMPLETE CATEGORY RANKINGS:

1. Technology
   → Profit: $145,455 (15.6% margin)
   → Sales: $836,154 (1,847 orders)
2. Office Supplies
   → Profit: $122,491 (13.8% margin)
   → Sales: $719,047 (6,026 orders)
3. Furniture
   → Profit: $18,451 (3.9% margin)
   → Sales: $742,000 (2,121 orders)

TOP INDIVIDUAL PRODUCT:

- Product: Canon imageCLASS 2200 Advanced Copier…
- Total Profit: $25,199.93
- Orders: 5

```
HBox(children=(Button(button_style='info', description='← Previous Question',␣
↪disabled=True, layout=Layout(hei…
```