1) GROSS MARGIN

total\_revenue = 127804.31

cogs = 96321.56

gross\_margin = ((total\_revenue - cogs) / total\_revenue) \* 100

print(f"Gross Margin: {gross\_margin:.2f}%")

OUTPUT

Gross Margin: 24.63%

2) THE MOST PROFITABLE VENDOR

vendors = [

{'name': 'Vendor 1', 'revenue': 6912.74, 'costs': 5357.15},

{'name': 'Vendor 2', 'revenue': 6704.47, 'costs': 5407.49},

{'name': 'Vendor 3', 'revenue': 23138.27, 'costs': 16767.6},

{'name': 'Vendor 4', 'revenue': 27255.22, 'costs': 20351.91},

{'name': 'Vendor 5', 'revenue': 7489.760, 'costs': 5873.62},

{'name': 'Vendor 6', 'revenue': 5487.87, 'costs': 4269.92},

{'name': 'Vendor 7', 'revenue': 21789.76, 'costs': 16828.71},

{'name': 'Vendor 8', 'revenue': 1393.84, 'costs': 1140.1},

{'name': 'Vendor 9', 'revenue': 16191.75, 'costs': 11925.05},

{'name': 'Vendor 10', 'revenue': 11440.63,'costs': 8400.01},

]

for vendor in vendors:

vendor['profit'] = vendor['revenue'] - vendor['costs']

most\_profitable\_vendor = max(vendors, key=lambda x: x['profit'])

print(f"The most profitable vendor is {most\_profitable\_vendor['name']} with a profit of ${most\_profitable\_vendor['profit']}")

OUTPUT

The most profitable vendor is Vendor 4 with a profit of $6903.310000000001

3) the least profitable vendor

for vendor in vendors:

vendor['profit'] = vendor['revenue'] - vendor['costs']

least\_profitable\_vendor = min (vendors, key=lambda x: x['profit'])

print(f"The least profitable vendor is {least\_profitable\_vendor['name']} with a profit of ${least\_profitable\_vendor['profit']}")

OUTPUT

The least profitable vendor is Vendor 8 with a profit of $253.74

4) the most profitable day

data = [

{'day': 'Monday', 'profit': 6847.34},

{'day': 'Tuesday', 'profit': 3305.35},

{'day': 'Wednesday', 'profit': 4392.47},

{'day': 'Thursday', 'profit': 4866.03},

{'day': 'Friday', 'profit': 3761.94},

{'day': 'Saturday', 'profit': 5223.16},

{'day': 'Sunday', 'profit': 3086.46},

]

day\_profits = {}

for entry in data:

day = entry['day']

profit = entry['profit']

day\_profits[day] = day\_profits.get(day, 0) + profit

most\_profitable\_day = max(day\_profits, key=day\_profits.get)

print(f"The most profitable day of the week is {most\_profitable\_day} with a total profit of ${day\_profits[most\_profitable\_day]}")

OUTPUT

The most profitable day of the week is Monday with a total profit of $6847.34

5) the least profitable day

day\_profits = {}

for entry in data:

day = entry['day']

profit = entry['profit']

day\_profits[day] = day\_profits.get(day, 0) + profit

least\_profitable\_day = min(day\_profits, key=day\_profits.get)

print(f"The least profitable day of the week is {least\_profitable\_day} with a total profit of ${day\_profits[least\_profitable\_day]}")

OUTPUT

The least profitable day of the week is SUNDAY

6) Can you calculate a 3-day average of the daily gross profit from the sales data.

YES.We can calculate the 3-day average of the daily gross profit from the sales data

daily\_profit = [6847.34, 3305.35, 4392.47, 4866.03, 3761.94, 5223.16, 3086.46]

three\_day\_averages = []

for i in range(2, len(daily\_profit)):

three\_day\_average = (daily\_profit[i] + daily\_profit[i - 1] + daily\_profit[i - 2]) / 3

three\_day\_averages.append(three\_day\_average)

for day, average in enumerate(three\_day\_averages, start=3):

print(f"3-day Average for Day {day}: ${average:.2f}")

output

3-day Average for Day 3: $4848.39

3-day Average for Day 4: $4187.95

3-day Average for Day 5: $4340.15

3-day Average for Day 6: $4617.04

3-day Average for Day 7: $4023.85

7)

daily\_profit = [6847.34, 3305.35, 4392.47, 4866.03, 3761.94, 5223.16, 3086.46]

three\_day\_averages = []

for i in range(2, len(daily\_profit)):

three\_day\_average = sum(daily\_profit[i-2:i+1]) / 3

three\_day\_averages.append(three\_day\_average)

for day, average in enumerate(three\_day\_averages, start=3):

trend = "Increasing" if average > three\_day\_averages[day - 4] else "Decreasing"

print(f"3-day Average for Day {day}: ${average:.2f} ({trend} trend)")

OUTPUT

3-day Average for Day 3: $4848.39 (Increasing trend)

3-day Average for Day 4: $4187.95 (Decreasing trend)

3-day Average for Day 5: $4340.15 (Increasing trend)

3-day Average for Day 6: $4617.04 (Increasing trend)

3-day Average for Day 7: $4023.85 (Decreasing trend)

This shows that the trend is fluctuating with the following data.

8)

AS THE VENOR HAS THE LOW RATING THE VENDOR HAS TO GET FIRED TO ASSESS THE INCREASE IN THE OPTIMIZATION OF THE GROSS PROFIT.

9) buying\_prices = [ 72.11,

75.08,

83.93,

52.01,

98.91,

63.34,

74.34,

98.11,

20.95,

88.45,

27.71,

10.48,

52.63,

69.41,

80.15,

92.7,

89.79,

62.22,

67.21,

58.42,

55.24,

71.96,

70.65,

40.63,

27.75,

10.19,

35.85,

93.29,

64.52,

50.49,

59.3,

14.76,

26.29,

18.66,

99.23,

93.63,

24.75,

17.24,

14.3,

15.42,

17.82,

59.01,

17.62,

53.89,

42.84,

92.54,

11.13,

10.31,

87.36,

17.85,

32.9,

94.29,

66.39,

78.7,

82.77,

67.04,

38.79,

48.76,

64.69,

28.9,

57.2,

57.37,

68.27,

83.76,

77.18,

64.82,

88.79,

63.3,

94.06,

56.28]

selling\_prices = [91.58,

83.71,

103.76,

64.53,

140.92,

85.13,

106.93,

146.76,

24.48,

102.31,

32.9,

14.31,

70.75,

80.31,

109.71,

132.53,

111.92,

69.25,

78.5,

68.74,

72.22,

105.37,

95.07,

53.71,

30.86,

11.65,

40.31,

135.33,

78.53,

59.99,

79.45,

16.25,

38.53,

22.45,

131.98,

119.13,

31.49,

19.39,

18.81,

22.79,

22.13,

84.3,

22.11,

64.51,

63.24,

129.69,

14.78,

12.26,

124.13,

20.47,

44.05,

113.3,

88.23,

92.87,

124.13,

83.95,

44.29,

59.41,

97.02,

36.83,

63.08,

72.03,

100.62,

96.1,

104.27,

87.45,

105.59,

92.92,

125.67,

82.16,

]

def calculate\_percentile(data, percentile):

sorted\_data = sorted(data)

index = int(percentile / 100 \* (len(sorted\_data) - 1))

return sorted\_data[index]

buying\_percentiles = [calculate\_percentile(buying\_prices, p) for p in [25, 50, 75]]

selling\_percentiles = [calculate\_percentile(selling\_prices, p) for p in [25, 50, 75]]

print("Percentiles for Buying Prices:")

print(f"25th Percentile: {buying\_percentiles[0]}")

print(f"50th Percentile (Median): {buying\_percentiles[1]}")

print(f"75th Percentile: {buying\_percentiles[2]}\n")

print("Percentiles for Selling Prices:")

print(f"25th Percentile: {selling\_percentiles[0]}")

print(f"50th Percentile (Median): {selling\_percentiles[1]}")

print(f"75th Percentile: {selling\_percentiles[2]}")

OUTPUT

Percentiles for Buying Prices:

25th Percentile: 28.9

50th Percentile (Median): 59.3

75th Percentile: 77.18

Percentiles for Selling Prices:

25th Percentile: 38.53

50th Percentile (Median): 78.53

75th Percentile: 102.31

10)

percentile\_25 = 25

percentile\_75 = 75

def identify\_outlier\_prices(prices, percentile\_25, percentile\_75):

lower\_threshold = calculate\_percentile(prices, percentile\_25)

upper\_threshold = calculate\_percentile(prices, percentile\_75)

outliers = [price for price in prices if price < lower\_threshold or price > upper\_threshold]

return outliers

def calculate\_percentile(data, percentile):

sorted\_data = sorted(data)

index = int(percentile / 100 \* (len(sorted\_data) - 1))

return sorted\_data[index]

outlier\_buying\_prices = identify\_outlier\_prices(buying\_prices, percentile\_25, percentile\_75)

outlier\_selling\_prices = identify\_outlier\_prices(selling\_prices, percentile\_25, percentile\_75)

print("Outlier Buying Prices:")

print(outlier\_buying\_prices)

print("\nOutlier Selling Prices:")

print(outlier\_selling\_prices)

OUTPUT

Outlier Buying Prices:

[83.93, 98.91, 98.11, 20.95, 88.45, 27.71, 10.48, 80.15, 92.7, 89.79, 27.75, 10.19, 93.29, 14.76, 26.29, 18.66, 99.23, 93.63, 24.75, 17.24, 14.3, 15.42, 17.82, 17.62, 92.54, 11.13, 10.31, 87.36, 17.85, 94.29, 78.7, 82.77, 83.76, 88.79, 94.06]

Outlier Selling Prices:

[103.76, 140.92, 106.93, 146.76, 24.48, 32.9, 14.31, 109.71, 132.53, 111.92, 105.37, 30.86, 11.65, 135.33, 16.25, 22.45, 131.98, 119.13, 31.49, 19.39, 18.81, 22.79, 22.13, 22.11, 129.69, 14.78, 12.26, 124.13, 20.47, 113.3, 124.13, 36.83, 104.27, 105.59, 125.67]

By analysing the data'S OUTLIER it shows Sapota’s prices fall below the 25th percentile or above the 75th percentile, potentially indicating they are buying at a high price or selling at a low price relative to their own transaction history.

11)"Vendor 4 has to be honored" suggests that SYA needs to carry out its duties or promises to vendor 4. With a profit of 6903.31, vendor 4 is the most lucrative.Hence, based on the cost of goods sold, fulfilling vendor 4 would have an impact on your COGS, which is a crucial factor in determining your gross margin. We need that you purchase goods and services for them at a specific cost. If vendor 4 provides competitive pricing or favorable conditions, your gross margin can increase.

12)Low cost of goods sold (COGS), quality and dependability, supplier exclusivity, demand and price power, long-term cooperation, supply chain efficiency, market circumstances, and economies of scale are all qualities that the company should look for in a vendor.

((Selling Price-COGS)/Selling Price) × 100% = Gross Margin

Thus, after taking into account a number of factors, we decide to partner with vendor 6 in order to achieve the maximum gross margin possible.

Vendor 6 has set a notional maximum margin of 68.69072% that the company may have from that branch.

13)

If you increase the price of Sapota by Rs. 1, and the quantity demanded decreases by, for example, 2%, you would have a percentage change in price (

P) of +1% and a percentage change in quantity demanded (

Q) of -2%.

If you decrease the price of Sapota by Rs. 1, and the quantity demanded increases by, for example, 3%, you would have a percentage change in price (

P) of -1% and a percentage change in quantity demanded (

Q) of +3%.

Using these values, you can calculate the price elasticity of Sapota for that branch as follows:

Price Elasticity of Sapota=% Change in Quantity Demanded / % Change in Price

Price Elasticity of Sapota= % Change in Price/ % Change in Quantity Demanded

?

Price Elasticity of Sapota=-2% /1%

=-2

Price Elasticity of Sapota= 1%/ -2%

?=-2

This indicates that Sapota's pricing elasticity for that branch is -2. Practically speaking, Sapota appears to be relatively inelastic, indicating that changes in its price have less of an impact on the quantity demanded, with a price elasticity of -2. Quantity demanded decreases by 2% for every 1% increase in price, whereas quantity demanded increases by 2% for every 1% decrease in price. Put differently, customers in this branch are less receptive to changes in Sapota's price.