Exploratory Data Analysis

Understanding the dataset to explore how the data is present in the database and if there is a need of creating some aggregated tables that can help with:

Vendor selection for profitability

Product Pricing Optimization

In [1]:
```
import pandas as pd
import sqlite3
```

In [2]:
```
# creating database connection
conn = sqlite3.connect('inventory.db')
```

In [3]:
```
# creating tables present in the database
tables = pd.read_sql_query("SELECT name FROM sqlite_master WHERE type ='table'
tables
```

Out[3]:

	name
0	begin_inventory
1	end_inventory
2	purchases
3	purchase_prices
4	sales
5	vendor_invoice

In [4]:
```
for table in tables ['name']:
    print('-'*50, f'{table}','-'*50)
    print("Count of records:", pd.read_sql(f"select count(*) as count from {ta
    display(pd.read_sql(f"select * from {table} limit 5", conn))
```

```
-------------------------------------------------- begin_inventory
--------------------------------------------------
Count of records: 619587
```

	InventoryId	Store	City	Brand	Description	Size	onHand
0	1_HARDERSFIELD_58	1	HARDERSFIELD	58	Gekkeikan Black & Gold Sake	750mL	8
1	1_HARDERSFIELD_60	1	HARDERSFIELD	60	Canadian Club 1858 VAP	750mL	7
2	1_HARDERSFIELD_62	1	HARDERSFIELD	62	Herradura Silver Tequila	750mL	6
3	1_HARDERSFIELD_63	1	HARDERSFIELD	63	Herradura Reposado Tequila	750mL	3
4	1_HARDERSFIELD_72	1	HARDERSFIELD	72	No. 3 London Dry Gin	750mL	6

```
------------------------------------------------- end_inventory
-------------------------------------------------
Count of records: 673467
```

	InventoryId	Store	City	Brand	Description	Size	onHand
0	1_HARDERSFIELD_58	1	HARDERSFIELD	58	Gekkeikan Black & Gold Sake	750mL	11
1	1_HARDERSFIELD_62	1	HARDERSFIELD	62	Herradura Silver Tequila	750mL	7
2	1_HARDERSFIELD_63	1	HARDERSFIELD	63	Herradura Reposado Tequila	750mL	7
3	1_HARDERSFIELD_72	1	HARDERSFIELD	72	No. 3 London Dry Gin	750mL	4
4	1_HARDERSFIELD_75	1	HARDERSFIELD	75	Three Olives Tomato Vodka	750mL	7

```
------------------------------------------------- purchases
-------------------------------------------------
Count of records: 7117422
```

	InventoryId	Store	Brand	Description	Size	VendorNumber	Vendo
0	69_MOUNTMEND_8412	69	8412	Tequila Ocho Plata Fresno	750mL	105	A BRAN
1	30_CULCHETH_5255	30	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	AM V BE
2	34_PITMERDEN_5215	34	5215	TGI Fridays Long Island Iced	1.75L	4466	AM V BE
3	1_HARDERSFIELD_5255	1	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	AM V BE
4	76_DONCASTER_2034	76	2034	Glendalough Double Barrel	750mL	388	AT IMP CO

---------- purchase_prices ----------

Count of records: 36783

	Brand	Description	Price	Size	Volume	Classification	PurchasePrice	Vendo
0	58	Gekkeikan Black & Gold Sake	12.99	750mL	750	1	9.28	
1	62	Herradura Silver Tequila	36.99	750mL	750	1	28.67	
2	63	Herradura Reposado Tequila	38.99	750mL	750	1	30.46	
3	72	No. 3 London Dry Gin	34.99	750mL	750	1	26.11	
4	75	Three Olives Tomato Vodka	14.99	750mL	750	1	10.94	

---------- sales ----------

Count of records: 25650726

	InventoryId	Store	Brand	Description	Size	SalesQuantity	SalesD
0	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glasses	750mL	1	
1	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glasses	750mL	2	
2	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glasses	750mL	1	
3	1_HARDERSFIELD_1004	1	1004	Jim Beam w/2 Rocks Glasses	750mL	1	
4	1_HARDERSFIELD_1005	1	1005	Maker's Mark Combo Pack	375mL 2 Pk	2	

```
------------------------------------------------- vendor_invoice
-------------------------------------------------
Count of records: 11086
```

	VendorNumber	VendorName	InvoiceDate	PONumber	PODate	PayDate
0	105	ALTAMAR BRANDS LLC	2024-01-04	8124	2023-12-21	2024-02-16
1	4466	AMERICAN VINTAGE BEVERAGE	2024-01-07	8137	2023-12-22	2024-02-21
2	388	ATLANTIC IMPORTING COMPANY	2024-01-09	8169	2023-12-24	2024-02-16
3	480	BACARDI USA INC	2024-01-12	8106	2023-12-20	2024-02-05
4	516	BANFI PRODUCTS CORP	2024-01-07	8170	2023-12-24	2024-02-12

In [5]:
```
purchases = pd.read_sql_query("Select * from purchases where VendorNumber = 44
purchases
```

Out[5]:

	InventoryId	Store	Brand	Description	Size	VendorNumber	V
0	30_CULCHETH_5255	30	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	
1	34_PITMERDEN_5215	34	5215	TGI Fridays Long Island Iced	1.75L	4466	
2	1_HARDERSFIELD_5255	1	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	
3	38_GOULCREST_5215	38	5215	TGI Fridays Long Island Iced	1.75L	4466	
4	59_CLAETHORPES_5215	59	5215	TGI Fridays Long Island Iced	1.75L	4466	
...	...	...	...	...	...	...	
6571	81_PEMBROKE_5215	81	5215	TGI Fridays Long Island Iced	1.75L	4466	
6572	62_KILMARNOCK_5255	62	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	
6573	34_PITMERDEN_5215	34	5215	TGI Fridays Long Island Iced	1.75L	4466	
6574	6_GOULCREST_5215	6	5215	TGI Fridays Long Island Iced	1.75L	4466	
6575	35_HALIVAARA_5255	35	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	

6576 rows × 16 columns

In [6]:
```
purchase_prices = pd.read_sql_query("Select * from purchase_prices where Vendo
purchase_prices
```

Out[6]:

	Brand	Description	Price	Size	Volume	Classification	PurchasePrice	Ve
0	5215	TGI Fridays Long Island Iced	12.99	1750mL	1750	1	9.41	
1	5255	TGI Fridays Ultimte Mudslide	12.99	1750mL	1750	1	9.35	
2	3140	TGI Fridays Orange Dream	14.99	1750mL	1750	1	11.19	
3	5215	TGI Fridays Long Island Iced	12.99	1750mL	1750	1	9.41	
4	5255	TGI Fridays Ultimte Mudslide	12.99	1750mL	1750	1	9.35	
5	3140	TGI Fridays Orange Dream	14.99	1750mL	1750	1	11.19	
6	5215	TGI Fridays Long Island Iced	12.99	1750mL	1750	1	9.41	
7	5255	TGI Fridays Ultimte Mudslide	12.99	1750mL	1750	1	9.35	
8	3140	TGI Fridays Orange Dream	14.99	1750mL	1750	1	11.19	

In [7]:
```
sales = pd.read_sql_query("Select * from sales where VendorNo = 4466",conn)
sales
```

Out[7]:

	InventoryId	Store	Brand	Description	Size	SalesQuantity	S
0	1_HARDERSFIELD_5215	1	5215	TGI Fridays Long Island Iced	1.75L	1	
1	1_HARDERSFIELD_5215	1	5215	TGI Fridays Long Island Iced	1.75L	1	
2	1_HARDERSFIELD_5215	1	5215	TGI Fridays Long Island Iced	1.75L	1	
3	1_HARDERSFIELD_5215	1	5215	TGI Fridays Long Island Iced	1.75L	1	
4	1_HARDERSFIELD_5215	1	5215	TGI Fridays Long Island Iced	1.75L	1	
...	...	...	...	...	...	...	
18901	9_BLACKPOOL_5215	9	5215	TGI Fridays Long Island Iced	1.75L	1	
18902	9_BLACKPOOL_5255	9	5255	TGI Fridays Ultimte Mudslide	1.75L	1	
18903	9_BLACKPOOL_5255	9	5255	TGI Fridays Ultimte Mudslide	1.75L	1	
18904	9_BLACKPOOL_5255	9	5255	TGI Fridays Ultimte Mudslide	1.75L	1	
18905	9_BLACKPOOL_5255	9	5255	TGI Fridays Ultimte Mudslide	1.75L	1	

18906 rows × 14 columns

In [8]:
```
purchases.groupby(['Brand','PurchasePrice'])[['Quantity','Dollars']].sum()
```

Out[8]:

		Quantity	Dollars
Brand	**PurchasePrice**		
3140	**11.19**	13920	155764.80
5215	**9.41**	14769	138976.29
5255	**9.35**	18645	174330.75

In [9]:
```
purchase_prices
```

Out[9]:

	Brand	Description	Price	Size	Volume	Classification	PurchasePrice	Ve
0	5215	TGI Fridays Long Island Iced	12.99	1750mL	1750	1	9.41	
1	5255	TGI Fridays Ultimte Mudslide	12.99	1750mL	1750	1	9.35	
2	3140	TGI Fridays Orange Dream	14.99	1750mL	1750	1	11.19	
3	5215	TGI Fridays Long Island Iced	12.99	1750mL	1750	1	9.41	
4	5255	TGI Fridays Ultimte Mudslide	12.99	1750mL	1750	1	9.35	
5	3140	TGI Fridays Orange Dream	14.99	1750mL	1750	1	11.19	
6	5215	TGI Fridays Long Island Iced	12.99	1750mL	1750	1	9.41	
7	5255	TGI Fridays Ultimte Mudslide	12.99	1750mL	1750	1	9.35	
8	3140	TGI Fridays Orange Dream	14.99	1750mL	1750	1	11.19	

In [10]:
```
conn = sqlite3.connect("inventory.db")
vendor_invoice = pd.read_sql("SELECT * FROM vendor_invoice", conn)
print(vendor_invoice.head())
```

```
   VendorNumber                  VendorName InvoiceDate  PONumber  \
0           105  ALTAMAR BRANDS LLC           2024-01-04      8124
1          4466  AMERICAN VINTAGE BEVERAGE    2024-01-07      8137
2           388  ATLANTIC IMPORTING COMPANY   2024-01-09      8169
3           480  BACARDI USA INC              2024-01-12      8106
4           516  BANFI PRODUCTS CORP          2024-01-07      8170

       PODate     PayDate  Quantity    Dollars  Freight Approval
0  2023-12-21  2024-02-16         6     214.26     3.47     None
1  2023-12-22  2024-02-21        15     140.55     8.57     None
2  2023-12-24  2024-02-16         5     106.60     4.61     None
3  2023-12-20  2024-02-05     10100  137483.78  2935.20     None
4  2023-12-24  2024-02-12      1935   15527.25   429.20     None
```

In [11]:
```
vendor_invoice['PONumber'].nunique()
```

Out[11]:
```
5543
```

In [12]:
```
vendor_invoice.columns
```

Out[12]:
```
Index(['VendorNumber', 'VendorName', 'InvoiceDate', 'PONumber', 'PODate',
       'PayDate', 'Quantity', 'Dollars', 'Freight', 'Approval'],
      dtype='object')
```

In [13]:
```
purchases
```

Out[13]:

	InventoryId	Store	Brand	Description	Size	VendorNumber	V
0	30_CULCHETH_5255	30	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	
1	34_PITMERDEN_5215	34	5215	TGI Fridays Long Island Iced	1.75L	4466	
2	1_HARDERSFIELD_5255	1	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	
3	38_GOULCREST_5215	38	5215	TGI Fridays Long Island Iced	1.75L	4466	
4	59_CLAETHORPES_5215	59	5215	TGI Fridays Long Island Iced	1.75L	4466	
...	...	...	...	...	...	...	
6571	81_PEMBROKE_5215	81	5215	TGI Fridays Long Island Iced	1.75L	4466	
6572	62_KILMARNOCK_5255	62	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	
6573	34_PITMERDEN_5215	34	5215	TGI Fridays Long Island Iced	1.75L	4466	
6574	6_GOULCREST_5215	6	5215	TGI Fridays Long Island Iced	1.75L	4466	
6575	35_HALIVAARA_5255	35	5255	TGI Fridays Ultimte Mudslide	1.75L	4466	

6576 rows × 16 columns

In [14]:
```
sales.groupby(['Brand'])[['SalesDollars', 'SalesPrice', 'SalesQuantity']].sum(
```

Out[14]:

Brand	SalesDollars	SalesPrice	SalesQuantity
3140	101062.20	60143.70	7780
5215	120832.98	83084.04	9302
5255	158374.08	102361.20	12192

The purchases table contains actual purchases data, including the data of purchases, products (brands) purchased by vendors, the amount paid (in dollars), and the quantity purchased.

The purchases price column is driven from the purchases_prices table, which provides product wise actual and purchases pricrs. The combination of vendor and brand is unique in this table.

The vendor_invoice table aggragates data from the purchases table, summarizing quantity and dollar amounts, along with an additional column for freight. This table maintains uniqueness based on venfor and PO number.

The sales table captures actual sales transations, detailing the brands purchased by vendors, the quantity sold, the selling price, and the revenue earned.

As the data that we need for analysis distributed tables, we need to create a summary table containing.

Purchases transactions made by vendors.

Sales transactions data.

Freight costs for each vendors.

Actual products prices from vendors.

In [25]:
```
vendor_invoice.columns
```

Out[25]:
```
Index(['VendorNumber', 'VendorName', 'InvoiceDate', 'PONumber', 'PODate',
       'PayDate', 'Quantity', 'Dollars', 'Freight', 'Approval'],
      dtype='object')
```

In [26]:
```
freight_summary = pd.read_sql_query(""" select VendorNumber, sum(freight) as F
from vendor_invoice
```

```
group by VendorNumber""", conn)
```

In [27]:
```
freight_summary
```

Out[27]:

	VendorNumber	FreightCost
0	2	54.16
1	54	0.96
2	60	735.04
3	105	124.78
4	200	12.38
...	...	...
121	98450	1712.04
122	99166	260.18
123	172662	356.68
124	173357	405.00
125	201359	0.18

126 rows × 2 columns

In [18]:
```
pd.read_sql_query("""SELECT
                    p.VendorNumber,
                    p.VendorName,
                    p.Brand,
                    p.PurchasePrice,
                    pp.Volume,
                    pp.Price as ActualPrice,
                    SUM(p.Quantity) as TotalPurchaseQuantity,
                    SUM(p.Dollars) as TotalPriceDollars
                    FROM purchases p
                    JOIN purchase_prices pp
                    ON p.Brand = pp.Brand
                    where p.PurchasePrice > 0
                    GROUP BY p.VendorNumber, p.VendorName, p.Brand
                    ORDER BY TotalPriceDollars""", conn)
```

Out[18]:

	VendorNumber	VendorName	Brand	PurchasePrice	Volume	ActualPric
0	7245	PROXIMO SPIRITS INC.	3065	0.71	50	0.9
1	3960	DIAGEO NORTH AMERICA INC	6127	1.47	200	1.9
2	3924	HEAVEN HILL DISTILLERIES	9123	0.74	50	0.9
3	8004	SAZERAC CO INC	5683	0.39	50	0.4
4	9815	WINE GROUP INC	8527	1.32	750	4.9
...	...	...	...	...	...	..
10687	3960	DIAGEO NORTH AMERICA INC	3545	21.89	1750	29.9
10688	3960	DIAGEO NORTH AMERICA INC	4261	16.17	1750	22.9
10689	17035	PERNOD RICARD USA	8068	18.24	1750	24.9
10690	4425	MARTIGNETTI COMPANIES	3405	23.19	1750	28.9
10691	1128	BROWN-FORMAN CORP	1233	26.27	1750	36.9

10692 rows × 8 columns

In [19]:
```
sales.columns
```

Out[19]:
```
Index(['InventoryId', 'Store', 'Brand', 'Description', 'Size', 'SalesQuantit
y',
       'SalesDollars', 'SalesPrice', 'SalesDate', 'Volume', 'Classification',
       'ExciseTax', 'VendorNo', 'VendorName'],
      dtype='object')
```

In [20]:
```
pd.read_sql_query(""" select
                    VendorNo,
                    Brand,
                    sum(SalesDollars) as TotalSalesDollars,
                    sum(SalesPrice) as TotalSalesPrice,
                    sum(SalesQuantity) as TotalSalesQuantity,
                    sum(ExciseTax) as TotalExciseTax
                    from sales
                    group by VendorNo, Brand
```

```
order by TotalSalesDollars""", conn)
```

Out[20]:

	VendorNo	Brand	TotalSalesDollars	TotalSalesPrice	TotalSalesQuantity
0	8004	5287	1.96	1.96	4
1	9206	2773	1.98	1.98	2
2	3252	3933	3.96	1.98	4
3	3924	9123	3.96	1.98	4
4	10050	3623	3.96	3.96	4
...	...	...	...	...	...
11267	3960	3545	8446215.24	1091556.56	271676
11268	3960	4261	8951945.76	840100.02	400824
11269	17035	8068	9076241.20	922280.30	374280
11270	4425	3405	9638146.98	1123024.74	320494
11271	1128	1233	10203839.02	1345638.62	284098

11272 rows × 6 columns

In [30]:

```
vendor_sales_summary = pd.read_sql_query("""
WITH FreightSummary AS (
    SELECT
        VendorNumber,
        SUM(Freight) AS FreightCost
    FROM vendor_invoice
    GROUP BY VendorNumber
),

PurchaseSummary AS (
    SELECT
        p.VendorNumber,
        p.VendorName,
        p.Brand,
        p.Description,
        p.PurchasePrice,
        pp.Price AS ActualPrice,
        pp.Volume,
        SUM(p.Quantity) AS TotalPurchaseQuantity,
        SUM(p.Dollars) AS TotalPurchaseDollars
    FROM purchases p
    JOIN purchase_prices pp
        ON p.Brand = pp.Brand
    WHERE p.PurchasePrice > 0
    GROUP BY p.VendorNumber, p.VendorName, p.Brand, p.Description, p.PurchaseP
),

SalesSummary AS (
```

```
    SELECT
        VendorNo,
        Brand,
        SUM(SalesQuantity) AS TotalSalesQuantity,
        SUM(SalesDollars) AS TotalSalesDollars,
        SUM(SalesPrice) AS TotalSalesPrice,
        SUM(ExciseTax) AS TotalExciseTax
    FROM sales
    GROUP BY VendorNo, Brand
)

SELECT
    ps.VendorNumber,
    ps.VendorName,
    ps.Brand,
    ps.Description,
    ps.PurchasePrice,
    ps.ActualPrice,
    ps.Volume,
    ps.TotalPurchaseQuantity,
    ps.TotalPurchaseDollars,
    ss.TotalSalesQuantity,
    ss.TotalSalesDollars,
    ss.TotalSalesPrice,
    ss.TotalExciseTax,
    fs.FreightCost
FROM PurchaseSummary ps
LEFT JOIN SalesSummary ss
    ON ps.VendorNumber = ss.VendorNo
    AND ps.Brand = ss.Brand
LEFT JOIN FreightSummary fs
    ON ps.VendorNumber = fs.VendorNumber
ORDER BY ps.TotalPurchaseDollars DESC
""", conn)
```

In [31]:
```
vendor_sales_summary
```

Out[31]:

	VendorNumber	VendorName	Brand	Description	PurchasePrice	Actua
0	1128	BROWN-FORMAN CORP	1233	Jack Daniels No 7 Black	26.27	
1	4425	MARTIGNETTI COMPANIES	3405	Tito's Handmade Vodka	23.19	
2	17035	PERNOD RICARD USA	8068	Absolut 80 Proof	18.24	
3	3960	DIAGEO NORTH AMERICA INC	4261	Capt Morgan Spiced Rum	16.17	
4	3960	DIAGEO NORTH AMERICA INC	3545	Ketel One Vodka	21.89	
...	...	...	...	...	...	
10687	9815	WINE GROUP INC	8527	Concannon Glen Ellen Wh Zin	1.32	
10688	8004	SAZERAC CO INC	5683	Dr McGillicuddy's Apple Pie	0.39	
10689	3924	HEAVEN HILL DISTILLERIES	9123	Deep Eddy Vodka	0.74	
10690	3960	DIAGEO NORTH AMERICA INC	6127	The Club Strawbry Margarita	1.47	
10691	7245	PROXIMO SPIRITS INC.	3065	Three Olives Grape Vodka	0.71	

10692 rows × 14 columns

The query generates a vendors-wise sales and purchases summary, which is valuable for:

Performace Optimization

- the query involves heavy joins and aggregation on large datasets like sales and purchases.
- Storing the pre-aggregated results avoids repeated expensive computations
- Helps in analyzing sales, purchases, and pricing for different vendors

and brands.

- Future benefits of storing this data for faster dashboarding & reporting.
- Instead of running expensive queries each time, dashboards can fetch data quickely from vendors_sales_summary

In [42]:
```
vendor_sales_summary.dtypes
```

Out[42]:
```
VendorNumber               int64
VendorName                object
Brand                      int64
Description               object
PurchasePrice            float64
ActualPrice              float64
Volume                   float64
TotalPurchaseQuantity      int64
TotalPurchaseDollars     float64
TotalSalesQuantity       float64
TotalSalesDollars        float64
TotalSalesPrice          float64
TotalExciseTax           float64
FreightCost              float64
dtype: object
```

In [41]:
```
vendor_sales_summary.isnull().sum()
```

Out[41]:
```
VendorNumber             0
VendorName               0
Brand                    0
Description              0
PurchasePrice            0
ActualPrice              0
Volume                   0
TotalPurchaseQuantity    0
TotalPurchaseDollars     0
TotalSalesQuantity       0
TotalSalesDollars        0
TotalSalesPrice          0
TotalExciseTax           0
FreightCost              0
dtype: int64
```

In [40]:
```
vendor_sales_summary['VendorName'].unique()
```

Out[40]:
```
array(['BROWN-FORMAN CORP', 'MARTIGNETTI COMPANIES', 'PERNOD RICARD USA',
       'DIAGEO NORTH AMERICA INC', 'BACARDI USA INC',
       'JIM BEAM BRANDS COMPANY', 'MAJESTIC FINE WINES',
       'ULTRA BEVERAGE COMPANY LLP', 'STOLI GROUP,(USA) LLC',
       'PROXIMO SPIRITS INC.', 'MOET HENNESSY USA INC', 'CAMPARI AMERICA',
       'SAZERAC CO INC', 'CONSTELLATION BRANDS INC', 'M S WALKER INC',
       'SAZERAC NORTH AMERICA INC.', 'PALM BAY INTERNATIONAL INC',
       'REMY COINTREAU USA INC', 'SIDNEY FRANK IMPORTING CO',
       'E & J GALLO WINERY', 'WILLIAM GRANT & SONS INC',
       'HEAVEN HILL DISTILLERIES', 'DISARONNO INTERNATIONAL LLC',
       'EDRINGTON AMERICAS', 'CASTLE BRANDS CORP.',
       'SOUTHERN WINE & SPIRITS NE', 'STE MICHELLE WINE ESTATES',
       'TRINCHERO FAMILY ESTATES', 'MHW LTD', 'WINE GROUP INC',
       'PERFECTA WINES', 'LUXCO INC', 'TREASURY WINE ESTATES',
       'DIAGEO CHATEAU ESTATE WINES', 'SHAW ROSS INT L IMP LTD',
       'PINE STATE TRADING CO', 'PHILLIPS PRODUCTS CO.',
       'CALEDONIA SPIRITS INC', 'STATE WINE & SPIRITS',
       'KOBRAND CORPORATION', 'BANFI PRODUCTS CORP',
       'VINEYARD BRANDS INC', 'DELICATO VINEYARDS INC',
       'FABRIZIA SPIRITS LLC', 'DUGGANS DISTILLED PRODUCTS',
       'Serralles Usa LLC', 'SEA HAGG DISTILLERY LLC',
       'OLE SMOKY DISTILLERY LLC', 'VRANKEN AMERICA', 'KLIN SPIRITS LLC',
       'LAIRD & CO', 'ADAMBA IMPORTS INTL INC',
       'LATITUDE BEVERAGE COMPANY', 'FREDERICK WILDMAN & SONS',
       'MCCORMICK DISTILLING CO', 'CHARLES JACQUIN ET CIE INC',
       'WESTERN SPIRITS BEVERAGE CO', 'MARSALLE COMPANY',
       'AMERICAN VINTAGE BEVERAGE', 'MANGO BOTTLING INC',
       'SWEET BABY VINEYARD', 'NICHE W & S', 'LABELLE VYDS AND WINERY',
       'FLAG HILL WINERY & VINEYARD', 'SMOKY QUARTZ DISTILLERY LLC',
       'PREMIUM PORT WINES, INC.', 'Russian Standard Vodka',
       'Dunn Wine Brokers', 'WEIN BAUER INC', 'BULLY BOY DISTILLERS',
       'ATLANTIC IMPORTING COMPANY', 'PREMIER DISTRIBUTORS',
       'VINILANDIA USA', 'PARK STREET IMPORTS LLC', 'TAKARA SAKE USA INC',
       'SEA BREEZE CELLARS LLC', 'STARK BREWING COMPANY', 'TY KU LLC',
       'PSP WINES', 'TAMWORTH DISTILLING', 'ZORVINO VINEYARDS',
       'SOUTHERN GLAZERS W&S OF NE', 'HOOD RIVER DISTILLERS, Inc.',
       'CRUSH WINES', 'POVERTY LANE ORCHARDS', 'DJINN SPIRITS LLC',
       'MOONLIGHT MEADERY', 'TALL SHIP DISTILLERY LLC',
       'FORTUNE WINE BROKERS LLC', 'BLACK COVE BEVERAGES', 'VINEXTRA INC',
       'SURVILLE ENTERPRISES CORP', 'JEWELL TOWNE VINEYARDS',
       'SWEETWATER FARM', 'ALTAMAR BRANDS LLC', 'CANDIA VINEYARDS',
       'INCREDIBREW INC', 'ALISA CARR BEVERAGES',
       'STELLAR IMPORTING CO LLC', 'FULCHINO VINEYARD INC',
       'IRA GOLDMAN AND WILLIAMS, LLP', 'Circa Wines',
       'VINEDREA WINES LLC', 'BLACK PRINCE DISTILLERY INC',
       'VINEYARD BRANDS LLC', 'THE IMPORTED GRAPE LLC',
       'WALPOLE MTN VIEW WINERY', 'GILMANTON WINERY & VINEYARD',
       'HAUNTING WHISPER VYDS', 'STAR INDUSTRIES INC.',
       'LOYAL DOG WINERY', 'R.P.IMPORTS INC', 'THE PIERPONT GROUP LLC',
       'APPOLO VINEYARDS LLC', 'BLACK ROCK SPIRITS LLC',
       'CENTEUR IMPORTS LLC', 'HIGHLAND WINE MERCHANTS LLC',
       'AMERICAN SPIRITS EXCHANGE', 'UNCORKED', 'BRONCO WINE COMPANY',
       'MILTONS DISTRIBUTING CO', 'TRUETT HURST', 'LAUREATE IMPORTS CO',
       'FANTASY FINE WINES CORP', 'AAPER ALCOHOL & CHEMICAL CO',
```

```
       'SILVER MOUNTAIN CIDERS', 'CAPSTONE INTERNATIONAL',
       'FLAVOR ESSENCE INC'], dtype=object)
```

In [35]:
```
vendor_sales_summary['Description'].unique()
```

Out[35]:
```
array(['Jack Daniels No 7 Black', "Tito's Handmade Vodka",
       'Absolut 80 Proof', ..., 'Crown Royal Apple',
       'Concannon Glen Ellen Wh Zin', 'The Club Strawbry Margarita'],
      shape=(9651,), dtype=object)
```

In [36]:
```
vendor_sales_summary['Volume'] = vendor_sales_summary['Volume'].astype('float6
```

In [37]:
```
vendor_sales_summary.fillna(0, inplace = True)
```

In [39]:
```
vendor_sales_summary['VendorName'] = vendor_sales_summary['VendorName'].str.st
```

In [43]:
```
vendor_sales_summary['GrossProfit'] = vendor_sales_summary['TotalSalesDollars'
```

In [44]:
```
vendor_sales_summary
```

Out[44]:

	VendorNumber	VendorName	Brand	Description	PurchasePrice	Actua
0	1128	BROWN-FORMAN CORP	1233	Jack Daniels No 7 Black	26.27	
1	4425	MARTIGNETTI COMPANIES	3405	Tito's Handmade Vodka	23.19	
2	17035	PERNOD RICARD USA	8068	Absolut 80 Proof	18.24	
3	3960	DIAGEO NORTH AMERICA INC	4261	Capt Morgan Spiced Rum	16.17	
4	3960	DIAGEO NORTH AMERICA INC	3545	Ketel One Vodka	21.89	
...	...	...	...	...	...	
10687	9815	WINE GROUP INC	8527	Concannon Glen Ellen Wh Zin	1.32	
10688	8004	SAZERAC CO INC	5683	Dr McGillicuddy's Apple Pie	0.39	
10689	3924	HEAVEN HILL DISTILLERIES	9123	Deep Eddy Vodka	0.74	
10690	3960	DIAGEO NORTH AMERICA INC	6127	The Club Strawbry Margarita	1.47	
10691	7245	PROXIMO SPIRITS INC.	3065	Three Olives Grape Vodka	0.71	

10692 rows × 15 columns

In [45]:
```
vendor_sales_summary['GrossProfit'].min()
```

Out[45]:
```
np.float64(-24598223.999999996)
```

In [46]:
```
vendor_sales_summary['ProfitMargin'] = (vendor_sales_summary['GrossProfit'] /
```

In [47]:
```
vendor_sales_summary['StockTurnover'] = vendor_sales_summary['TotalSalesQuanti
```

In [48]:
```
vendor_sales_summary['SalestoPurchaseRatio'] = vendor_sales_summary['TotalSale
```

In [49]:
```
vendor_sales_summary.columns
```

Out[49]:
```
Index(['VendorNumber', 'VendorName', 'Brand', 'Description', 'PurchasePrice',
       'ActualPrice', 'Volume', 'TotalPurchaseQuantity',
       'TotalPurchaseDollars', 'TotalSalesQuantity', 'TotalSalesDollars',
       'TotalSalesPrice', 'TotalExciseTax', 'FreightCost', 'GrossProfit',
       'ProfitMargin', 'StockTurnover', 'SalestoPurchaseRatio'],
      dtype='object')
```

In [50]:
```
cursor = conn.cursor()
```

In [51]:
```
cursor.execute(""" create table vendor_sales_summary (
    VendorNumber int,
    VendorName varchar(100),
    Brand int
    Description varchar(100),
    PurchasePrice decimal(10, 2),
    ActualPrice decimal(10,2),
    Volume,
    TotalPurchaseQuantity int,
    TotalPurchaseDollars decimal(15,2),
    TotalSalesQuantity int,
    TotalSalesDollars decimal(15,2),
    TotalSalesPrice decimal(15,2),
    TotalExciseTax decimal(15,2),
    FreightCost decimal(15,2),
    GrossProfit decimal(15,2),
    ProfitMargin decimal(15,2),
    StockTurnover decimal(15,2),
    SalesToPurchaseRatio decimal(15,2),
    Primary key (VendorNumber, Brand)
);
""")
```

Out[51]:
```
<sqlite3.Cursor at 0x22712caa1c0>
```

In [54]:
```
pd.read_sql_query("select * from vendor_sales_summary", conn)
```

Out[54]:

	VendorNumber	VendorName	Brand	Description	PurchasePrice	Actua
0	1128	BROWN-FORMAN CORP	1233	Jack Daniels No 7 Black	26.27	
1	4425	MARTIGNETTI COMPANIES	3405	Tito's Handmade Vodka	23.19	
2	17035	PERNOD RICARD USA	8068	Absolut 80 Proof	18.24	
3	3960	DIAGEO NORTH AMERICA INC	4261	Capt Morgan Spiced Rum	16.17	
4	3960	DIAGEO NORTH AMERICA INC	3545	Ketel One Vodka	21.89	
...	...	...	...	...	...	
10687	9815	WINE GROUP INC	8527	Concannon Glen Ellen Wh Zin	1.32	
10688	8004	SAZERAC CO INC	5683	Dr McGillicuddy's Apple Pie	0.39	
10689	3924	HEAVEN HILL DISTILLERIES	9123	Deep Eddy Vodka	0.74	
10690	3960	DIAGEO NORTH AMERICA INC	6127	The Club Strawbry Margarita	1.47	
10691	7245	PROXIMO SPIRITS INC.	3065	Three Olives Grape Vodka	0.71	

10692 rows × 18 columns

In [65]:
```
vendor_sales_summary.to_sql('vendor_sales_summary', conn, if_exists = 'replace
```

Out[65]:
```
10692
```

In [78]:
```
import os
print(os.getcwd())  # Notebook ka current working directory
print(os.listdir("logs"))  # logs folder ke andar kya files hain
```

```
D:\Data Analysis\Project Data Vendor Performance P
['.ipynb_checkpoints', '.log', 'ingestion_db.log']
```

In [81]:
```
log_path = r"D:/Data Analysis/Project Data Vendor Performance P/logs\get_vendo
os.makedirs(os.path.dirname(log_path), exist_ok=True)

logging.basicConfig(
```

```
    filename=log_path,
    level=logging.DEBUG,
    format="%(asctime)s - %(levelname)s - %(message)s",
    filemode="a"
)
```

In [76]:

In [ ]:

In [66]:
```
import sqlite3
import pandas as pd
import logging
from ingestion_db import ingest_db

logging.basicConfig(
    filename="logs/get_vendor_summary.log",
    level=logging.DEBUG,
    format="%(asctime)s - %(levelname)s - %(message)s",
    filemode="a"
)

def create_vendor_summary(conn):
    """Merge different tables to get vendor summary and add new columns"""
    vendor_sales_summary = pd.read_sql_query("""
        WITH FreightSummary AS (
            SELECT
                VendorNumber,
                SUM(Freight) AS FreightCost
            FROM vendor_invoice
            GROUP BY VendorNumber
        ),

        PurchaseSummary AS (
            SELECT
                p.VendorNumber,
                p.VendorName,
                p.Brand,
                p.Description,
                p.PurchasePrice,
                pp.Price AS ActualPrice,
                pp.Volume,
                SUM(p.Quantity) AS TotalPurchaseQuantity,
                SUM(p.Dollars) AS TotalPurchaseDollars
            FROM purchases p
            JOIN purchase_prices pp
                ON p.Brand = pp.Brand
            WHERE p.PurchasePrice > 0
            GROUP BY p.VendorNumber, p.VendorName, p.Brand, p.Description, p.P
        ),

        SalesSummary AS (
            SELECT
```

```
                VendorNo,
                Brand,
                SUM(SalesQuantity) AS TotalSalesQuantity,
                SUM(SalesDollars) AS TotalSalesDollars,
                SUM(SalesPrice) AS TotalSalesPrice,
                SUM(ExciseTax) AS TotalExciseTax
            FROM sales
            GROUP BY VendorNo, Brand
        )

        SELECT
            ps.VendorNumber,
            ps.VendorName,
            ps.Brand,
            ps.Description,
            ps.PurchasePrice,
            ps.ActualPrice,
            ps.Volume,
            ps.TotalPurchaseQuantity,
            ps.TotalPurchaseDollars,
            ss.TotalSalesQuantity,
            ss.TotalSalesDollars,
            ss.TotalSalesPrice,
            ss.TotalExciseTax,
            fs.FreightCost
        FROM PurchaseSummary ps
        LEFT JOIN SalesSummary ss
            ON ps.VendorNumber = ss.VendorNo
            AND ps.Brand = ss.Brand
        LEFT JOIN FreightSummary fs
            ON ps.VendorNumber = fs.VendorNumber
        ORDER BY ps.TotalPurchaseDollars DESC""", conn)

    return vendor_sales_summary


def clean_data(df):
    """Clean the data"""
    # changing datatype
    df['Volume'] = df['Volume'].astype(float)

    # filling missing value with 0
    df.fillna(0, inplace=True)

    # stripping spaces
    df['VendorName'] = df['VendorName'].str.strip()
    df['Description'] = df['Description'].str.strip()

    # creating new columns
    df['GrossProfit'] = df['TotalSalesDollars'] - df['TotalPurchaseDollars']
    df['ProfitMargin'] = (df['GrossProfit'] / df['TotalSalesDollars']) * 100
    df['StockTurnover'] = df['TotalSalesQuantity'] / df['TotalPurchaseQuantity
    df['SalestoPurchaseRatio'] = df['TotalSalesDollars'] / df['TotalPurchaseDo
```

```
    return df


if __name__ == '__main__':
    conn = sqlite3.connect('inventory.db')

    logging.info('Creating Vendor Summary Table ....')
    summary_df = create_vendor_summary(conn)
    logging.info(summary_df.head())

    logging.info('Cleaning Data.....')
    clean_df = clean_data(summary_df)
    logging.info(clean_df.head())

    logging.info('Ingesting data.....')
    ingest_db(clean_df, 'vendor_sales_summary', conn)
    logging.info('Complete')
```

In [ ]:

```python
import sqlite3
import pandas as pd
import logging
from ingestion_db import ingest_db

logging.basicConfig(
    filename="logs/get_vendor_summary.log",
    level=logging.DEBUG,
    format="%(asctime)s - %(levelname)s - %(message)s",
    filemode="a"
)

def create_vendor_summary(conn):
    """Merge different tables to get vendor summary and add new columns"""
    vendor_sales_summary = pd.read_sql_query("""
        WITH FreightSummary AS (
            SELECT
                VendorNumber,
                SUM(Freight) AS FreightCost
            FROM vendor_invoice
            GROUP BY VendorNumber
        ),

        PurchaseSummary AS (
            SELECT
                p.VendorNumber,
                p.VendorName,
                p.Brand,
                p.Description,
                p.PurchasePrice,
                pp.Price AS ActualPrice,
                pp.Volume,
                SUM(p.Quantity) AS TotalPurchaseQuantity,
                SUM(p.Dollars) AS TotalPurchaseDollars
            FROM purchases p
            JOIN purchase_prices pp
                ON p.Brand = pp.Brand
            WHERE p.PurchasePrice > 0
            GROUP BY p.VendorNumber, p.VendorName, p.Brand, p.Description, p.PurchasePrice, pp.Price, pp.Volume
        ),

        SalesSummary AS (
            SELECT
                VendorNo,
                Brand,
                SUM(SalesQuantity) AS TotalSalesQuantity,
                SUM(SalesDollars) AS TotalSalesDollars,
                SUM(SalesPrice) AS TotalSalesPrice,
                SUM(ExciseTax) AS TotalExciseTax
            FROM sales
            GROUP BY VendorNo, Brand


        SELECT
            ps.VendorNumber,
            ps.VendorName,
            ps.Brand,
            ps.Description,
            ps.PurchasePrice,
            ps.ActualPrice,
            ps.Volume,
            ps.TotalPurchaseQuantity,
            ps.TotalPurchaseDollars,
            ss.TotalSalesQuantity,
            ss.TotalSalesDollars,
            ss.TotalSalesPrice,
            ss.TotalExciseTax,
            fs.FreightCost
        FROM PurchaseSummary ps
        LEFT JOIN SalesSummary ss
            ON ps.VendorNumber = ss.VendorNo
            AND ps.Brand = ss.Brand
        LEFT JOIN FreightSummary fs
            ON ps.VendorNumber = fs.VendorNumber
        ORDER BY ps.TotalPurchaseDollars DESC""", conn)

    return vendor_sales_summary


def clean_data(df):
    """Clean the data"""
    # changing datatype
    df['Volume'] = df['Volume'].astype(float)

    # filling missing value with 0
    df.fillna(0, inplace=True)

    # stripping spaces
    df['VendorName'] = df['VendorName'].str.strip()
    df['Description'] = df['Description'].str.strip()

    # creating new columns
    df['GrossProfit'] = df['TotalSalesDollars'] - df['TotalPurchaseDollars']
    df['ProfitMargin'] = (df['GrossProfit'] / df['TotalSalesDollars']) * 100
    df['StockTurnover'] = df['TotalSalesQuantity'] / df['TotalPurchaseQuantity']
    df['SalestoPurchaseRatio'] = df['TotalSalesDollars'] / df['TotalPurchaseDollars']

    return df


if __name__ == '__main__':
    conn = sqlite3.connect('inventory.db')

    logging.info('Creating Vendor Summary Table ....')
    summary_df = create_vendor_summary(conn)
    logging.info(summary_df.head())

    logging.info('Cleaning Data.....')
    clean_df = clean_data(summary_df)
    logging.info(clean_df.head())

    logging.info('Ingesting data.....')
    ingest_db(clean_df, 'vendor_sales_summary', conn)
    logging.info('Complete')
```

```python
import pandas as pd
import os
from sqlalchemy import create_engine
import logging
import time

logging.basicConfig( filename="logs/ingestion_db.log",
                     level = logging.DEBUG,
                     format = "%(asctime)s - %(levelname)s - %(message)s",
                     filemode = "a"
                   )


engine = create_engine('sqlite:///inventory.db')
def ingest_large_csv(file, table_name, engine, read_chunksize=100000, insert_chunksize=5000):


    for chunk in pd.read_csv(file, chunksize=read_chunksize):
        chunk.to_sql(
            table_name,
            con=engine,
            if_exists='append',
            index=False,
            chunksize=insert_chunksize
        )
        print(f"Inserted {len(chunk)} rows into {table_name}...")

def ingest_db(df, table_name, engine):
    ''' this function will ingest the dataframe into database table'''
    df.to_sql(table_name, con = engine, if_exists = 'replace', index = False)

def load_raw_data():
    ''' this function will load the CSVs as dataframe and ingest into db...'''
    start = time.time()
    for file in os.listdir("data"):
        if file.endswith(".csv"):
            logging.info(f"Ingesting {file} in db ")
            ingest_large_csv("data/"+file, file[:-4], engine)
    end = time.time()
    total_time = (end - start)/60
    logging.info('----------------Ingestion Complete--------------------')
    logging.info(f'\nTotal Time Taken: {total_time} minutes')
if __name__ =='__main__':
    load_raw_data()
```

In [1]:
```
import pandas as pd
import os
from sqlalchemy import create_engine
import logging
import time

logging.basicConfig( filename="logs/ingestion_db.log",
                     level = logging.DEBUG,
                     format = "%(asctime)s - %(levelname)s - %(message)s",
                     filemode = "a"
                   )


engine = create_engine('sqlite:///inventory.db')
def ingest_large_csv(file, table_name, engine, read_chunksize=100000, insert_c


    for chunk in pd.read_csv(file, chunksize=read_chunksize):
        chunk.to_sql(
            table_name,
            con=engine,
            if_exists='append',
            index=False,
            chunksize=insert_chunksize
        )
        print(f"Inserted {len(chunk)} rows into {table_name}...")

def ingest_db(df, table_name, engine):
    """ this function will ingest the dataframe into database table"""
    df.to_sql(table_name, con = engine, if_exists = 'replace', index = False)

def load_raw_data():
    """ this function will load the CSVs as dataframe and ingest into db... ""
    start = time.time()
    for file in os.listdir("data"):
        if file.endswith(".csv"):
            logging.info(f"Ingesting {file} in db ")
            ingest_large_csv("data/"+file, file[:-4], engine)
    end = time.time()
    total_time = (end - start)/60
    logging.info('----------------Ingestion Complete--------------------')
    logging.info(f'\nTotal Time Taken: {total_time} minutes')
if __name__ =='__main__':
    load_raw_data()
```

```
Inserted 100000 rows into begin_inventory...
Inserted 100000 rows into begin_inventory...
Inserted 6529 rows into begin_inventory...
Inserted 100000 rows into end_inventory...
Inserted 100000 rows into end_inventory...
Inserted 24489 rows into end_inventory...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 100000 rows into purchases...
Inserted 72474 rows into purchases...
Inserted 12261 rows into purchase_prices...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
```

```
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
```

```
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 100000 rows into sales...
Inserted 25363 rows into sales...
Inserted 5543 rows into vendor_invoice...
```

In [ ]:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import sqlite3
from scipy.stats import ttest_ind
import scipy.stats as stats
warnings.filterwarnings('ignore')
```

Loading the dataset

In [2]:
```
#creating database connection
conn = sqlite3.connect('inventory.db')

# fetching vendor summary data
df = pd.read_sql_query("select * from vendor_sales_summary", conn)
df.head()
```

Out[2]:

	VendorNumber	VendorName	Brand	Description	PurchasePrice	ActualPrice
0	1128	BROWN-FORMAN CORP	1233	Jack Daniels No 7 Black	26.27	36.99
1	4425	MARTIGNETTI COMPANIES	3405	Tito's Handmade Vodka	23.19	28.99
2	17035	PERNOD RICARD USA	8068	Absolut 80 Proof	18.24	24.99
3	3960	DIAGEO NORTH AMERICA INC	4261	Capt Morgan Spiced Rum	16.17	22.99
4	3960	DIAGEO NORTH AMERICA INC	3545	Ketel One Vodka	21.89	29.99

Explotary Data Analysis

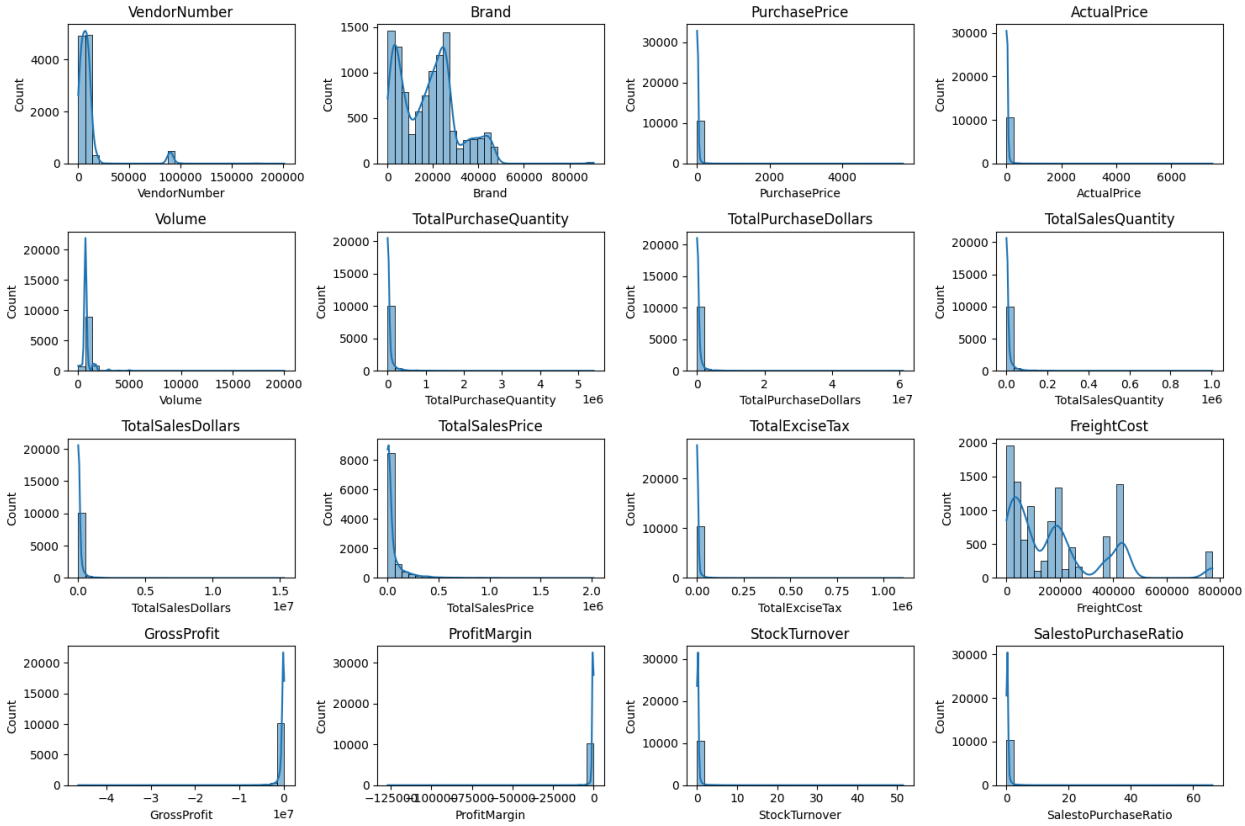
-previously, we examined the various tables in the database to identify key variables, understand their relationships, and determine which ones should be included in the final analysis. -In this phase of EDA, we will analyze the resultant table to gain insights into the distribution of each column. This will help us understand data patterns, identify anomalies, and ensure data quality before proceeding with further anlysis.

In [3]:
```
df.describe().T
#summary statistics
```

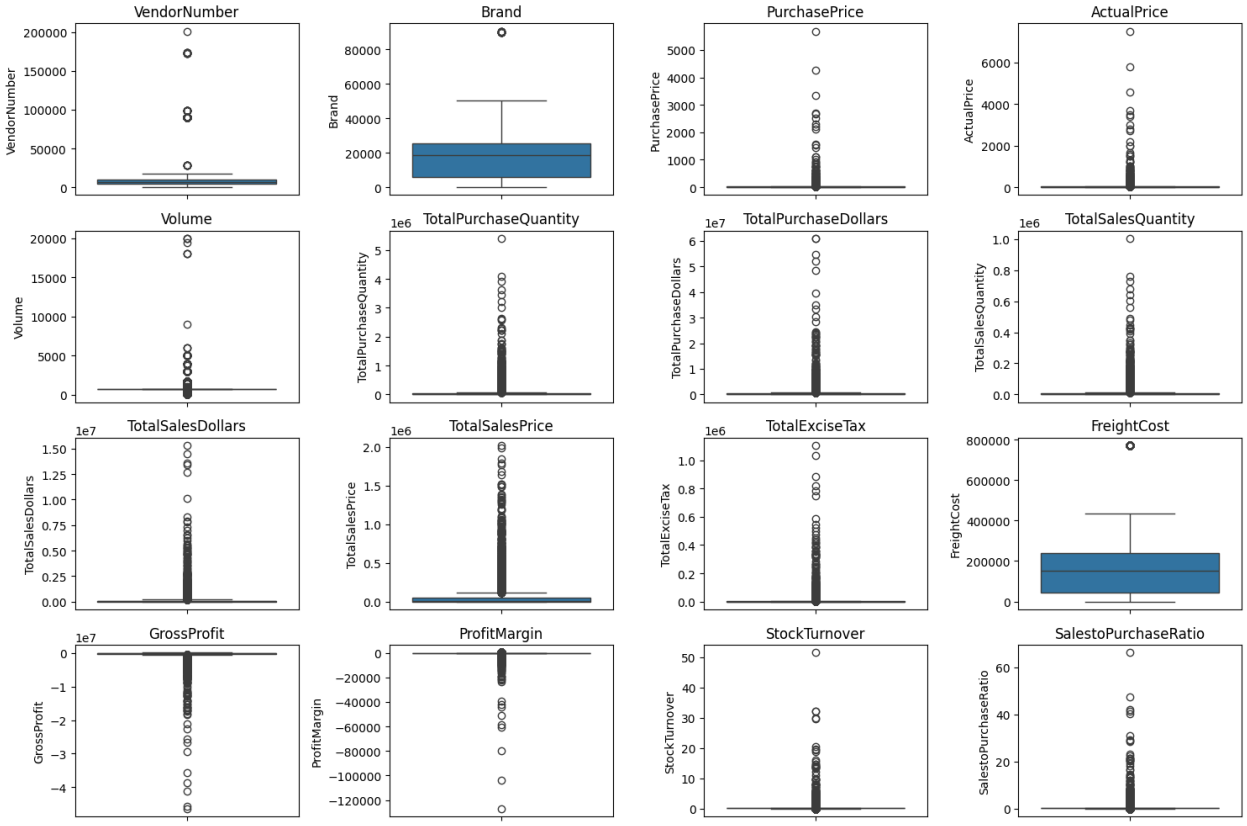
Out[3]:

	count	mean	std	min	
VendorNumber	10692.0	1.065065e+04	1.875352e+04	2.00	3
Brand	10692.0	1.803923e+04	1.266219e+04	58.00	5
PurchasePrice	10692.0	2.438530e+01	1.092694e+02	0.36	
ActualPrice	10692.0	3.564367e+01	1.482460e+02	0.49	
Volume	10692.0	8.473605e+02	6.643092e+02	50.00	
TotalPurchaseQuantity	10692.0	5.025419e+04	1.775214e+05	16.00	
TotalPurchaseDollars	10692.0	4.817071e+05	1.969085e+06	11.36	7
TotalSalesQuantity	10692.0	9.232446e+03	3.285855e+04	0.00	
TotalSalesDollars	10692.0	1.267172e+05	5.029658e+05	0.00	2
TotalSalesPrice	10692.0	5.638135e+04	1.348583e+05	0.00	
TotalExciseTax	10692.0	5.322679e+03	3.292675e+04	0.00	
FreightCost	10692.0	1.843013e+05	1.828154e+05	0.27	42
GrossProfit	10692.0	-3.549899e+05	1.468550e+06	-46407439.05	-244
ProfitMargin	10692.0	-inf	NaN	-inf	-
StockTurnover	10692.0	3.200238e-01	1.128836e+00	0.00	
SalestoPurchaseRatio	10692.0	4.695731e-01	1.586075e+00	0.00	

In [4]:
```
# distribution plots for numericals columns
numerical_cols = df.select_dtypes(include=np.number).columns

plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols):
    plt.subplot(4, 4, i+1) # adjust grid layout as needed
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(col)
plt.tight_layout()
plt.show()
```



In [5]:
```
#outliers detection with boxplots
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols):
    plt.subplot(4,4, i+1)
    sns.boxplot(y=df[col])
    plt.title(col)
plt.tight_layout()
plt.show()
```



summary Statistics insights:

Negative & Zero values:

- Gross profit: minimum value is -52,002.78, indicating losses. some products or transactions may be selling at a loss due to high costs or selling at discounts lower than the purchase price.
- Profit margin: has a minimum of -infinity, which suggests cases where revenue is zero or even lower than costs.
- Total sales quantity & sales dollars: minimum calues are 0, meaning some products were purchased but never sold. these could be slow-moving or obsolete stock.

Outliers indicated by high standard deviations:

- Purchases & actual prices: the max values(5,681.81, 7,499.99) are siginificantly higher than the mean (24.39& 35.64) indicating potential premium products.
- freight cost: huge variation, from (0.09 to 257,032.07), suggests

logistics inefficiencies or bulk shipments.

- stock turnover: ranges from 0 to 274.5, implying some products sell extremely fast while others remain in stock indefinitely. value more than 1 indicates that sold quantity for that product is higher than purchased quantity due to either sales are being fulfilled from older stock.

In [6]:
```
# let's filter the data by removing inconsistencies
df =pd.read_sql_query(""" select * from vendor_sales_summary
                          where GrossProfit >0
                          and ProfitMargin > 0
                          and TotalSalesQuantity > 0""", conn)
```

In [7]:
```
df
```

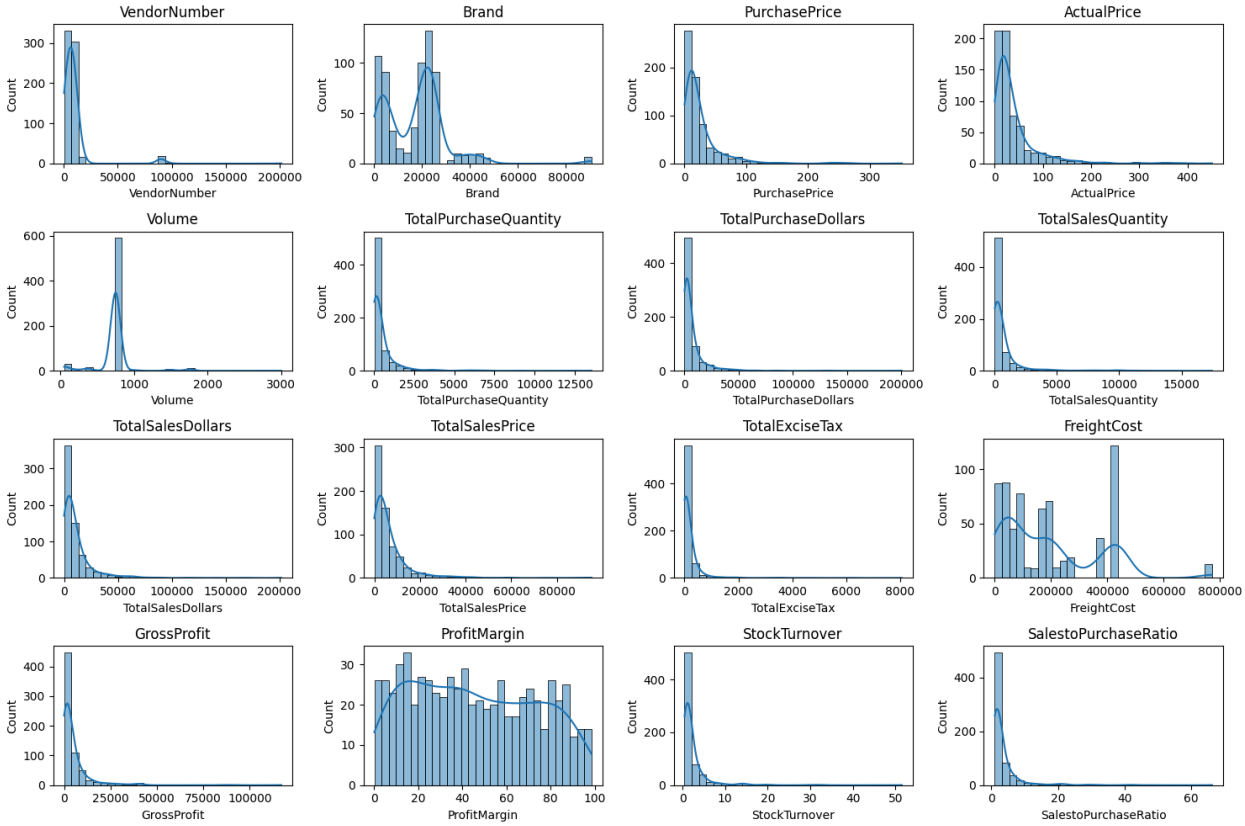
Out[7]:

	VendorNumber	VendorName	Brand	Description	PurchasePrice	Actual
0	480	BACARDI USA INC	4881	Bacardi Twin Pack 2/ 750mls	14.81	
1	2555	DISARONNO INTERNATIONAL LLC	1212	DiSaronna Amaretto Sour VAP	14.38	
2	10754	PERFECTA WINES	10264	Fort Ross Pnt Nr Sonoma Cst	19.60	
3	653	STATE WINE & SPIRITS	23256	Robert Hall Cab Svgn	9.39	
4	1128	BROWN-FORMAN CORP	1722	Jack Daniels Sinatra Century	351.55	4
...	...	...	...	...	...	
664	3960	DIAGEO NORTH AMERICA INC	2626	Crown Royal Apple	1.42	
665	9815	WINE GROUP INC	8527	Concannon Glen Ellen Wh Zin	1.32	
666	8004	SAZERAC CO INC	5683	Dr McGillicuddy's Apple Pie	0.39	
667	3960	DIAGEO NORTH AMERICA INC	6127	The Club Strawbry Margarita	1.47	
668	7245	PROXIMO SPIRITS INC.	3065	Three Olives Grape Vodka	0.71	

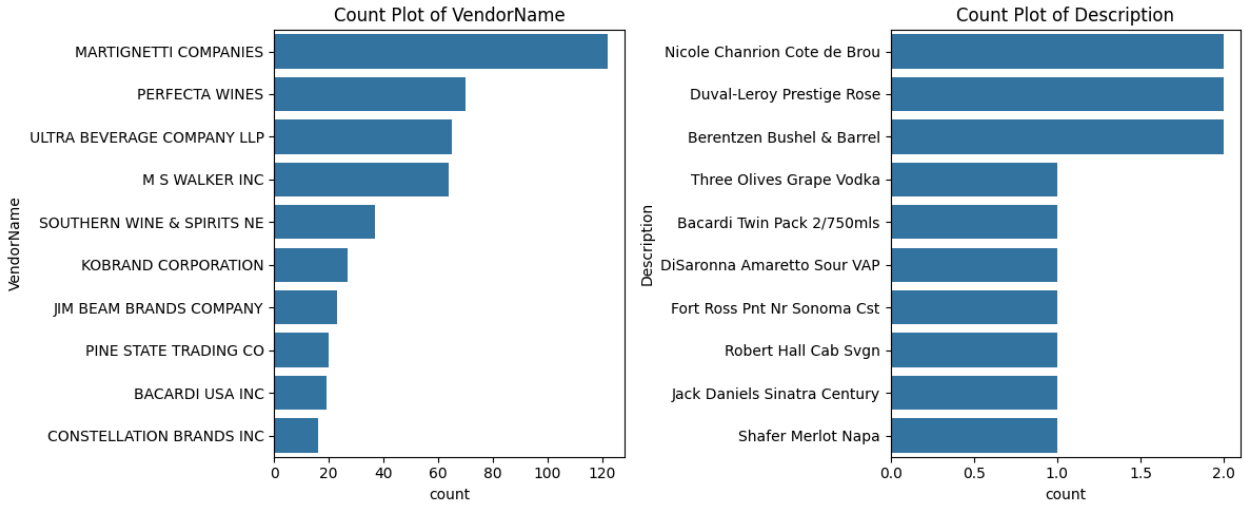
669 rows × 18 columns

In [8]:

```
# distribution plots for numericals columns
numerical_cols = df.select_dtypes(include=np.number).columns

plt.figure(figsize=(15, 10))
for i, col in enumerate (numerical_cols):
    plt.subplot(4, 4, i+1) # adjust grid layout as needed
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(col)
plt.tight_layout()
plt.show()
```

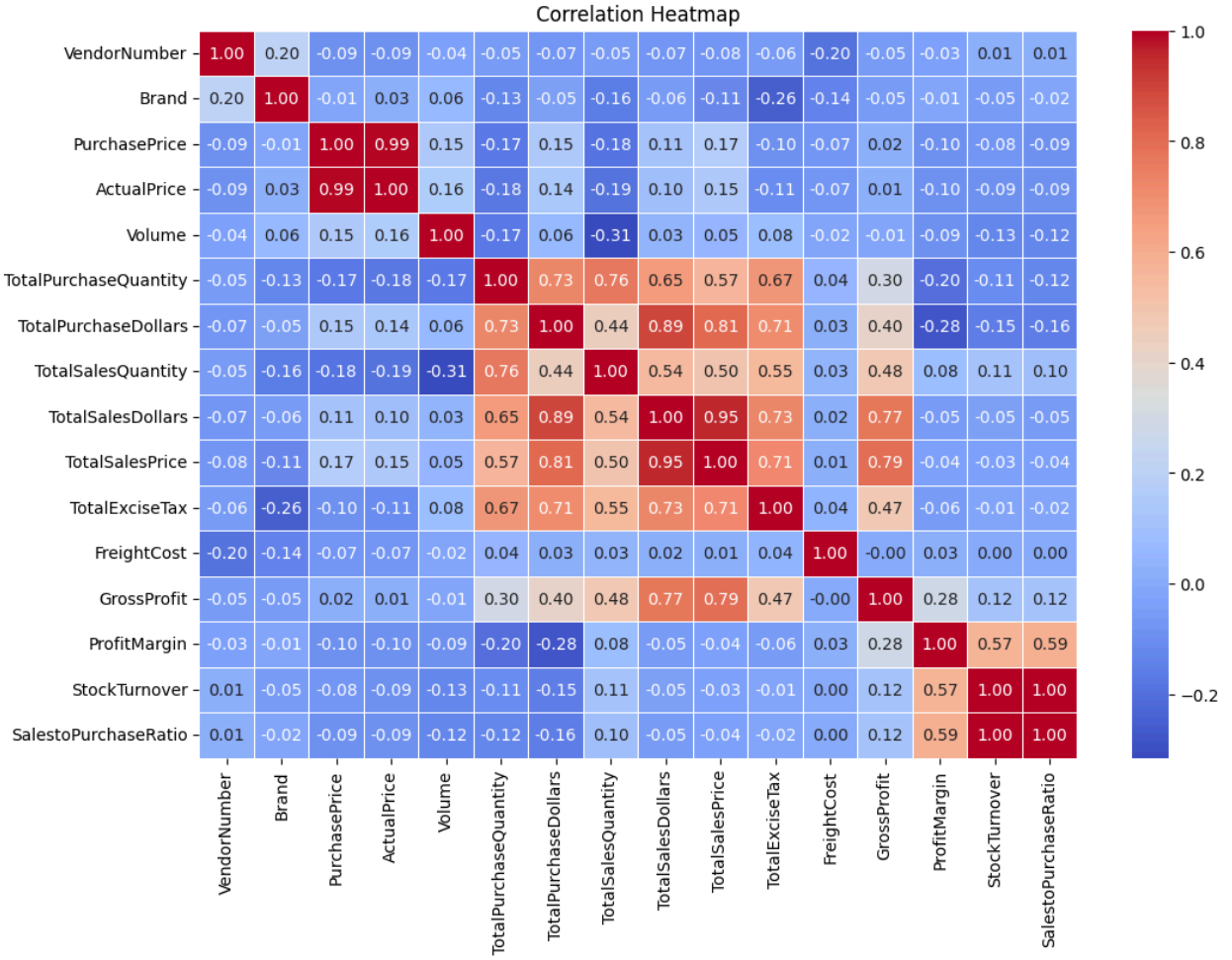


In [9]:
```
# count plots for categorical columns
categorical_cols = ["VendorName", "Description"]

plt.figure(figsize=(12, 5))
for i, col in enumerate(categorical_cols):
    plt.subplot(1, 2, i+1)
    sns.countplot(y=df[col], order=df[col].value_counts().index[:10]) # top 10
    plt.title(f"Count Plot of {col}")
plt.tight_layout()
plt.show()
```



In [10]:
```
# correlation heatmap
```

```
plt.figure(figsize=(12, 8))
correlation_matrix = df[numerical_cols].corr()
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", linewi
plt.title("Correlation Heatmap")
plt.show()
```



Correaltions Insights

- PurchasePrice has weak correaltions with totalsalesdollars (-0.012) and grossprofit (-0.016) , suggesting that price variations do not significantly impact sales revenue or profit.
- strong correlation between profit margin & total sales quantity (0.999) , confirming efficient inventory turnover.
- negative correlation between profit margin & total sales price (-0.179) suggests that a as sales price increases, margins decrease, possibly due to competitive pricing pressures.
- stock turnover has weak negative correlations with both grossprofit (-0.038) and profitmargin (-0.055) indicating that faster turnover does not necessarily result in higher profitability.

Data Analysis

- Identify brands that needs promotional or pricing adjustments which exhibit lower sales performance but higher profit margins.

In [13]:
```
brand_performance = df.groupby('Description').agg({
    'TotalSalesDollars':'sum',
    'ProfitMargin': 'mean'}).reset_index()
```

In [14]:
```
low_sales_threshold = brand_performance['TotalSalesDollars'].quantile(0.15)
high_margin_threshold = brand_performance['ProfitMargin'].quantile(0.85)
```

In [15]:
```
low_sales_threshold
```

Out[15]:
```
np.float64(1667.25)
```

In [16]:
```
high_margin_threshold
```

Out[16]:
```
np.float64(80.53179377674624)
```

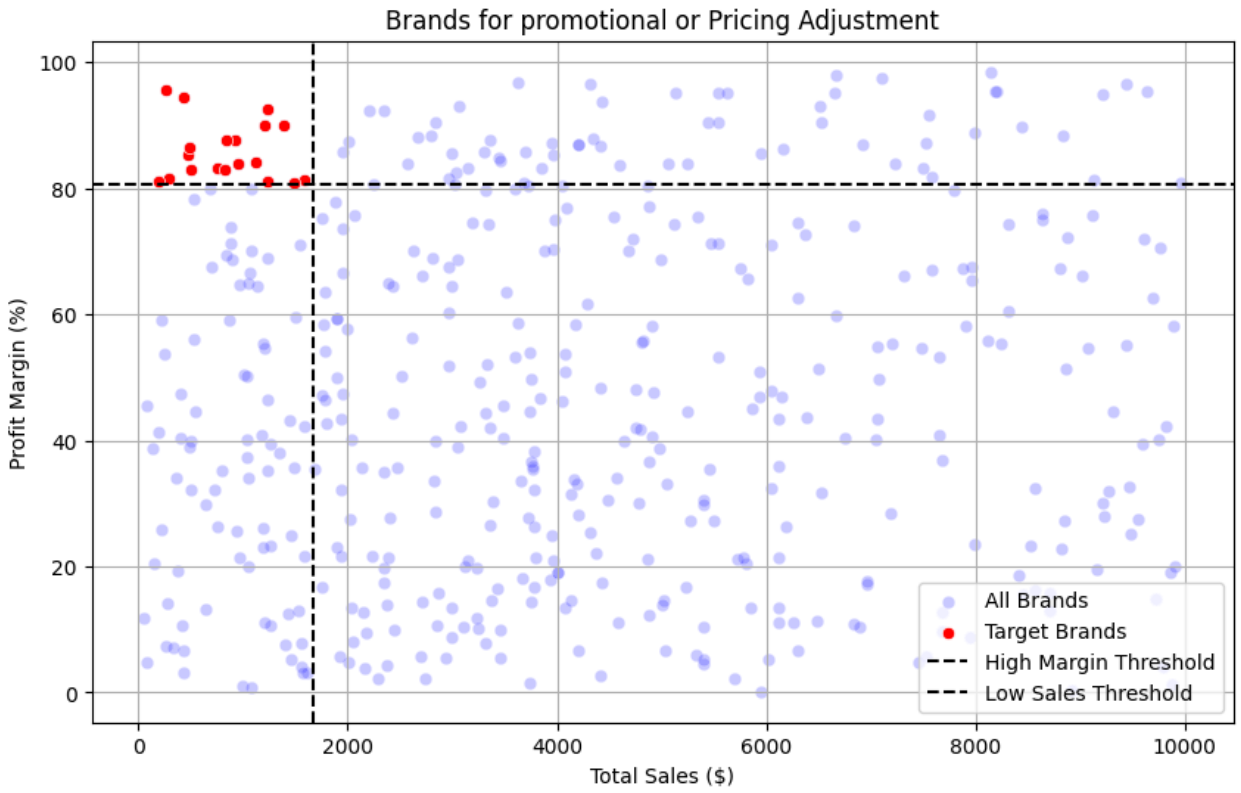
In [17]:
```
# filter brands with low sales but high profit margins
target_brands = brand_performance[
    (brand_performance['TotalSalesDollars'] <= low_sales_threshold) &
    (brand_performance['ProfitMargin'] >= high_margin_threshold)
    ]
print("Brands with low sales but high profit margins: ")
display(target_brands.sort_values('TotalSalesDollars'))
```

```
Brands with low sales but high profit margins:
```

	Description	TotalSalesDollars	ProfitMargin
267	Dr McGillicuddy's Apple Pie	196.98	80.992994
608	Three Olives Grape Vodka	255.42	95.552423
33	Aresti Pnt Nr Curico Vly	284.43	81.549063
600	The Club Strawbry Margarita	429.84	94.528197
481	Piehole Apple Pie	469.26	85.270426
515	Riunite Sweet White	485.19	86.479523
531	Sauza Sparkling Margarita	503.28	82.896201
251	DeKuyper Mixed Berry Medley	758.31	83.204758
642	Vigne A Porrona Rosso	818.37	83.068783
415	Mad Dogs & Englishmen Jumil	839.40	87.553014
0	12 Days of Pearls Gift Set	929.07	87.617725
437	Mojoshot Blue Lagoon RTD	950.40	84.023569
571	St Germain Liqueur	1124.25	84.202802
50	Bacardi Oakheart Spiced Trav	1198.80	90.043377
48	Bacardi Limon Traveler	1228.77	81.015162
570	St Elder Elderflower Liqueur	1232.55	92.626668
215	Chi Chi's Chocolate Malt RTD	1384.74	89.924462
159	Ch La Fleur Patris-Querre 09	1481.43	80.775332
21	Altadonna Vermentino	1591.23	81.297487

In [18]:
```
brand_performance = brand_performance[brand_performance['TotalSalesDollars']<1
```

In [19]:
```
plt.figure(figsize=(10,6))
sns.scatterplot(data=brand_performance, x='TotalSalesDollars', y='ProfitMargir
sns.scatterplot(data=target_brands, x='TotalSalesDollars', y='ProfitMargin', c

plt.axhline(high_margin_threshold, linestyle='--', color='black', label="High
plt.axvline(low_sales_threshold, linestyle='--', color='black', label="Low Sal

plt.xlabel("Total Sales ($)")
plt.ylabel("Profit Margin (%)")
plt.title("Brands for promotional or Pricing Adjustment")
plt.legend()
plt.grid(True)
plt.show()
```



Which vendors and brands demonstrate the highest sales performance

In [20]:
```
def format_dollars(value):
    if value>=1_000_000:
        return f"{value / 1_000_000:.2f}M"
    elif value>=1_000:
        return f"{value / 1_000: .2f}K"
    else:
        return str(value)
```

In [53]:
```
# top vendors & brands by sales performance
top_vendors = df.groupby("VendorName")["TotalSalesDollars"].sum().nlargest(10)
top_brands = df.groupby("Description")["TotalSalesDollars"].sum().nlargest(10)
top_vendors
```

Out[53]:
```
VendorName
MARTIGNETTI COMPANIES          1476319.59
ULTRA BEVERAGE COMPANY LLP      966800.94
M S WALKER INC                  677901.96
BACARDI USA INC                 547922.16
PERFECTA WINES                  537688.74
SOUTHERN WINE & SPIRITS NE      423435.87
BROWN-FORMAN CORP               365208.69
STATE WINE & SPIRITS            217284.45
DISARONNO INTERNATIONAL LLC     185169.15
PERNOD RICARD USA               173913.69
Name: TotalSalesDollars, dtype: float64
```

In [46]:
```
top_brands
```

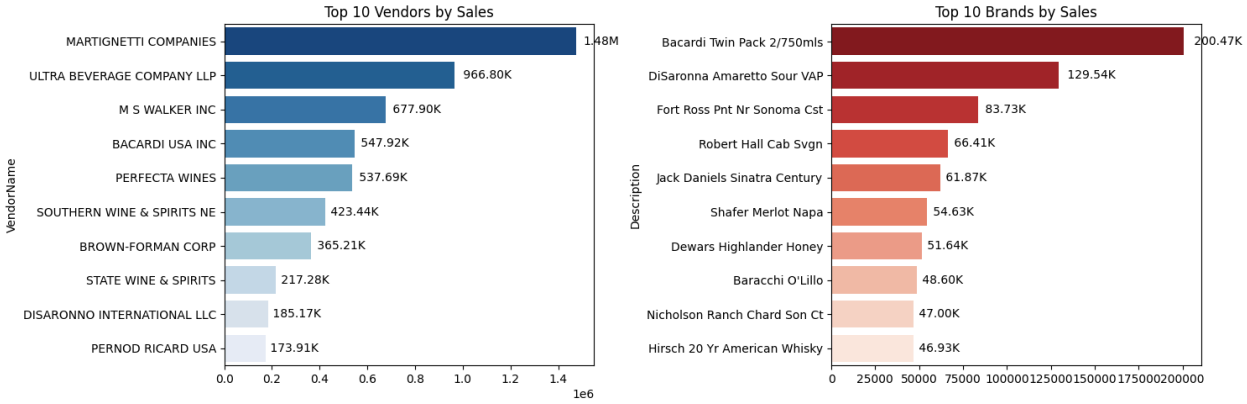
Out[46]:
```
Description
Bacardi Twin Pack 2/750mls      200468.16
DiSaronna Amaretto Sour VAP     129535.04
Fort Ross Pnt Nr Sonoma Cst      83731.20
Robert Hall Cab Svgn             66406.08
Jack Daniels Sinatra Century     61872.80
Shafer Merlot Napa               54628.80
Dewars Highlander Honey          51636.48
Baracchi O'Lillo                 48600.00
Nicholson Ranch Chard Son Ct     46996.32
Hirsch 20 Yr American Whisky     46931.52
Name: TotalPurchaseDollars, dtype: float64
```

In [47]:
```
top_brands.apply(lambda x: format_dollars(x))
```

Out[47]:
```
Description
Bacardi Twin Pack 2/750mls       200.47K
DiSaronna Amaretto Sour VAP      129.54K
Fort Ross Pnt Nr Sonoma Cst       83.73K
Robert Hall Cab Svgn              66.41K
Jack Daniels Sinatra Century      61.87K
Shafer Merlot Napa                54.63K
Dewars Highlander Honey           51.64K
Baracchi O'Lillo                  48.60K
Nicholson Ranch Chard Son Ct      47.00K
Hirsch 20 Yr American Whisky      46.93K
Name: TotalPurchaseDollars, dtype: object
```

In [48]:
```
plt.figure(figsize=(15, 5))

#  🔹 Plot for top vendors (Barplot)
plt.subplot(1, 2, 1)
ax1 = sns.barplot(y=top_vendors.index.astype(str), x=top_vendors.values, palet
plt.title("Top 10 Vendors by Sales")

# Add value labels
for bar in ax1.patches:
    ax1.text(
        bar.get_width() + (bar.get_width() * 0.02),
        bar.get_y() + bar.get_height() / 2,
        format_dollars(bar.get_width()),
        ha='left', va='center', fontsize=10, color='black')

#  🔹 Plot for top brands (Barplot)
plt.subplot(1, 2, 2)
ax2 = sns.barplot(y=top_brands.index.astype(str), x=top_brands.values, palette
plt.title("Top 10 Brands by Sales")

# Add value labels
for bar in ax2.patches:
    ax2.text(
        bar.get_width() + (bar.get_width() * 0.02),
```

```
        bar.get_y() + bar.get_height() / 2,
        format_dollars(bar.get_width()),
        ha='left', va='center', fontsize=10, color='black')

plt.tight_layout()
plt.show()
```



Which vendors contrubute the most to total purchases dollars

In [150…

```
df.groupby('VendorName').agg({
    'TotalSalesDollars':'sum',
    'GrossProfit':'sum',
    'TotalPurchaseDollars':'sum',
}).reset_index()
```

Out[150…

	VendorName	TotalSalesDollars	GrossProfit	TotalPurchaseDollars
0	ALISA CARR BEVERAGES	126806.04	31234.20	95571.84
1	ATLANTIC IMPORTING COMPANY	11366.31	1109.03	10257.28
2	BACARDI USA INC	547922.16	207461.36	340460.80
3	BANFI PRODUCTS CORP	14442.36	5243.48	9198.88
4	BROWN-FORMAN CORP	365208.69	191207.89	174000.80
...	...	...	...	...
57	VINILANDIA USA	78212.13	47681.41	30530.72
58	VRANKEN AMERICA	8378.19	5495.15	2883.04
59	WESTERN SPIRITS BEVERAGE CO	4977.51	1930.47	3047.04
60	WILLIAM GRANT & SONS INC	53669.58	15360.14	38309.44
61	WINE GROUP INC	14204.67	7878.59	6326.08

62 rows × 4 columns

In [151…
```
vendor_performance = df.groupby('VendorName').agg({
    'TotalPurchaseDollars':'sum',
    'GrossProfit':'sum',
    'TotalSalesDollars': 'sum'
}).reset_index()
vendor_performance.shape
```

Out[151… (62, 4)

In [154…
```
print(vendor_performance.columns)
```

```
Index(['VendorName', 'TotalPurchaseDollars', 'GrossProfit',
       'TotalSalesDollars', 'Purchase_Contribution%'],
      dtype='object')
```

In [153…
```
vendor_performance['Purchase_Contribution%'] = vendor_performance['TotalPurcha
```

In [155…
```
round(vendor_performance.sort_values('Purchase_Contribution%', ascending=False
```

Out[155…

	VendorName	TotalPurchaseDollars	GrossProfit	TotalSalesDollars	Purchase
29	MARTIGNETTI COMPANIES	881010.40	595309.19	1476319.59	
55	ULTRA BEVERAGE COMPANY LLP	571004.80	395796.14	966800.94	
2	BACARDI USA INC	340460.80	207461.36	547922.16	
36	PERFECTA WINES	310116.32	227572.42	537688.74	
27	M S WALKER INC	308977.92	368924.04	677901.96	
...	...	...	...	...	
13	DUGGANS DISTILLED PRODUCTS	556.32	85.29	641.61	
34	OLE SMOKY DISTILLERY LLC	402.88	28.85	431.73	
17	FLAVOR ESSENCE INC	272.00	4151.23	4423.23	
9	Circa Wines	257.60	581.80	839.40	
45	SAZERAC NORTH AMERICA INC.	139.52	1245.22	1384.74	

62 rows × 5 columns

In [156…

```
# display top 10 vendors
top_vendors = vendor_performance.head(10)
top_vendors ['TotalSalesDollars'] = top_vendors['TotalSalesDollars'].apply(for
top_vendors ['TotalPurchaseDollars'] = top_vendors['TotalPurchaseDollars'].app
top_vendors ['GrossProfit'] = top_vendors['GrossProfit'].apply(format_dollars)
top_vendors
```

Out[156…

	VendorName	TotalPurchaseDollars	GrossProfit	TotalSalesDollars	Purchase
0	ALISA CARR BEVERAGES	95.57K	31.23K	126.81K	
1	ATLANTIC IMPORTING COMPANY	10.26K	1.11K	11.37K	
2	BACARDI USA INC	340.46K	207.46K	547.92K	
3	BANFI PRODUCTS CORP	9.20K	5.24K	14.44K	
4	BROWN-FORMAN CORP	174.00K	191.21K	365.21K	
5	CAMPARI AMERICA	40.62K	13.05K	53.68K	
6	CASTLE BRANDS CORP.	13.38K	7.01K	20.39K	
7	CONSTELLATION BRANDS INC	99.65K	55.92K	155.57K	
8	CRUSH WINES	10.99K	6.88K	17.87K	
9	Circa Wines	257.6	581.8	839.4	

In [157…
```
top_vendors['Purchase_Contribution%'].sum()
```

Out[157…
```
np.float64(17.966480848825473)
```

In [160…
```
print(top_vendors.columns)
```

```
Index(['VendorName', 'TotalPurchaseDollars', 'GrossProfit',
       'TotalSalesDollars', 'Purchase_Contribution%',
       'Cumulative_Contribution%'],
      dtype='object')
```

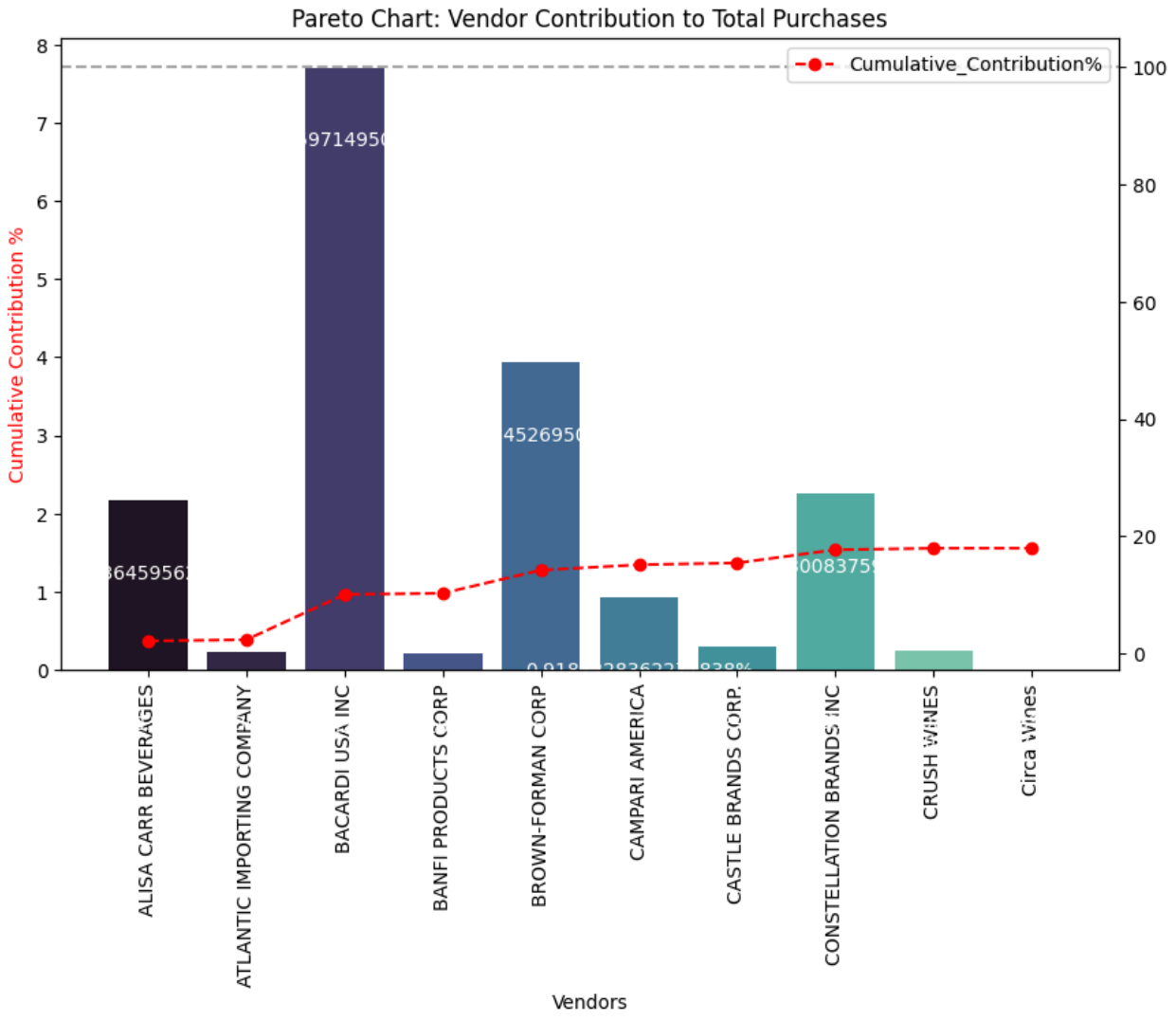
In [159…
```
top_vendors['Cumulative_Contribution%'] = top_vendors['Purchase_Contribution%'
top_vendors
```

Out[159…

	VendorName	TotalPurchaseDollars	GrossProfit	TotalSalesDollars	Purchase
0	ALISA CARR BEVERAGES	95.57K	31.23K	126.81K	
1	ATLANTIC IMPORTING COMPANY	10.26K	1.11K	11.37K	
2	BACARDI USA INC	340.46K	207.46K	547.92K	
3	BANFI PRODUCTS CORP	9.20K	5.24K	14.44K	
4	BROWN-FORMAN CORP	174.00K	191.21K	365.21K	
5	CAMPARI AMERICA	40.62K	13.05K	53.68K	
6	CASTLE BRANDS CORP.	13.38K	7.01K	20.39K	
7	CONSTELLATION BRANDS INC	99.65K	55.92K	155.57K	
8	CRUSH WINES	10.99K	6.88K	17.87K	
9	Circa Wines	257.6	581.8	839.4	

In [161…

```
top_vendors['Cumulative_Contribution%'] = top_vendors['Purchase_Contribution%'

fig, ax1 = plt.subplots(figsize=(10, 6))

# Bar plot for purchase contrubution%
sns.barplot(x=top_vendors['VendorName'], y=top_vendors['Purchase_Contribution%

for i, value in enumerate(top_vendors['Purchase_Contribution%']):
    ax1.text(i, value-1, str(value)+'%', ha='center', fontsize=10, color='whit

#line plot for cumulative contribution%
ax2 = ax1.twinx()
ax2.plot(top_vendors['VendorName'], top_vendors['Cumulative_Contribution%'], c

ax1.set_xticklabels(top_vendors['VendorName'], rotation=90)
ax1.set_ylabel('Purchase Contribution %', color='blue')
ax1.set_ylabel('Cumulative Contribution %', color='red')
ax1.set_xlabel('Vendors')
ax1.set_title('Pareto Chart: Vendor Contribution to Total Purchases')

ax2.axhline(y=100, color='gray', linestyle='dashed', alpha=0.7)
ax2.legend(loc='upper right')

plt.show()
```



How much to total procurement is dependent on the top vendors?

In [162…
```
print(f"Total Purchase Contribution of top 10 vendors is {round(top_vendors['P
```
```
Total Purchase Contribution of top 10 vendors is 17.97 %
```

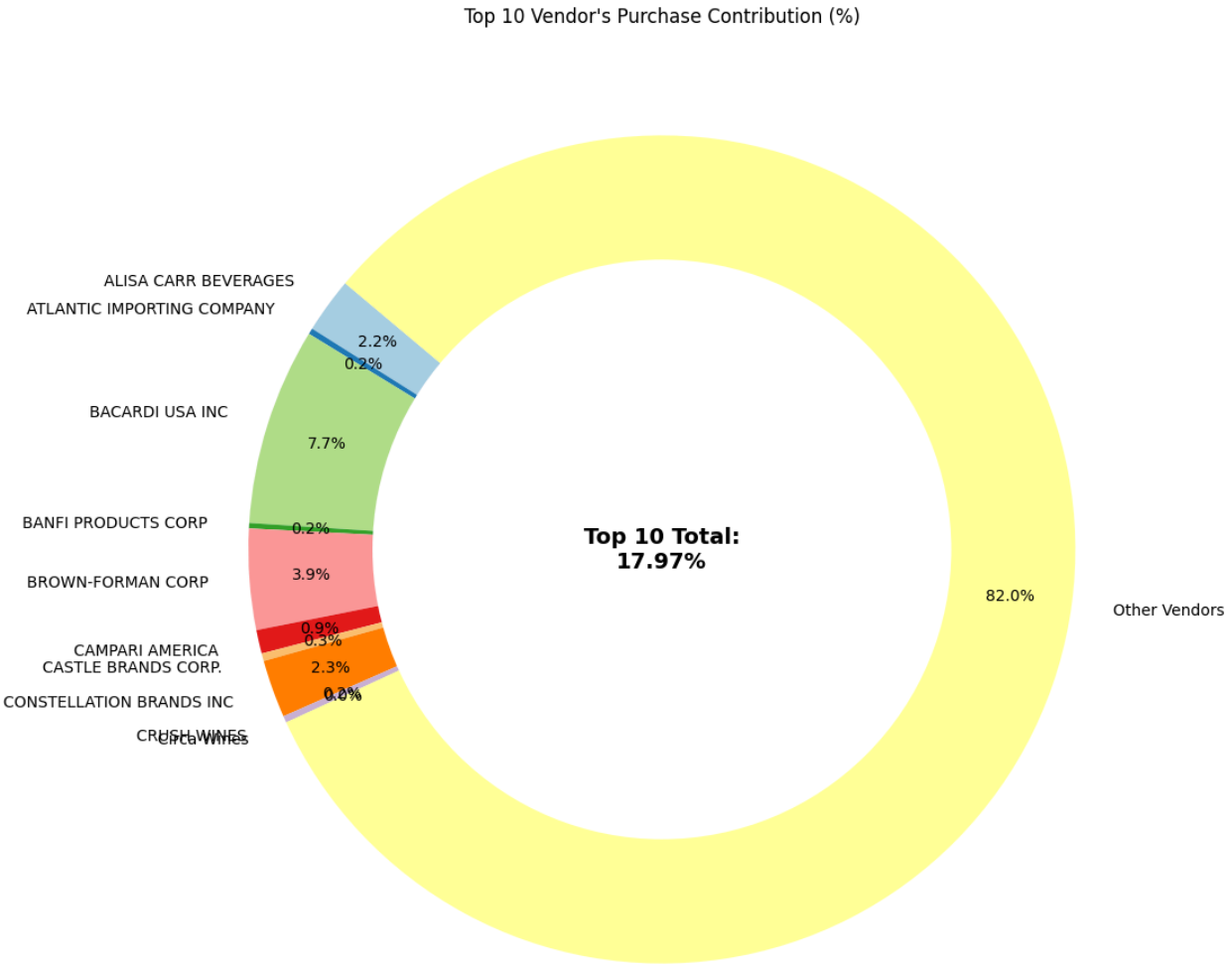
In [163…
```
print(type(top_vendors))
```
```
<class 'pandas.core.frame.DataFrame'>
```

In [164…
```
print(top_vendors.columns)
```
```
Index(['VendorName', 'TotalPurchaseDollars', 'GrossProfit',
       'TotalSalesDollars', 'Purchase_Contribution%',
       'Cumulative_Contribution%'],
      dtype='object')
```

In [177…
```
import matplotlib.pyplot as plt
```

```python

# Yahan pehle check karo actual columns
print(top_vendors.columns)

# Correct column name use karo (agar naam 'PurchaseContribution%' hai to neech
vendors = list(top_vendors['VendorName'].values)
purchase_contributions = list(top_vendors['Purchase_Contribution%'].values)  #

# Contribution calculate
total_contribution = sum(purchase_contributions)
remaining_contribution = 100 - total_contribution

# Append "Other Vendors" category
vendors.append("Other Vendors")
purchase_contributions.append(remaining_contribution)

# Donut Chart
fig, ax = plt.subplots(figsize=(12, 12))
wedges, texts, autotexts = ax.pie(
    purchase_contributions,
    labels=vendors,
    autopct='%1.1f%%',
    startangle=140,
    pctdistance=0.85,
    colors=plt.cm.Paired.colors
)

# Draw a white circle in the center to create a "donut" effect.
centre_circle = plt.Circle((0, 0), 0.7, fc='white')
fig.gca().add_artist(centre_circle)

# Add Total Contribution annotation in the centre
plt.text(
    0, 0,
    f"Top 10 Total:\n{total_contribution:.2f}%",
    fontsize=14,
    fontweight='bold',
    ha='center',
    va='center'
)

plt.title("Top 10 Vendor's Purchase Contribution (%)")
plt.show()
```

```
Index(['VendorName', 'TotalPurchaseDollars', 'GrossProfit',
       'TotalSalesDollars', 'Purchase_Contribution%',
       'Cumulative_Contribution%'],
      dtype='object')
```



Does purchasing in bulk reduce the unit price, and what is the optimal purchase volume for cost savings?

In [112…
```
df['UnitPurchasePrice'] = df['TotalPurchaseDollars'] / df['TotalPurchaseQuanti
```

In [114…
```
df["OrderSize"] = pd.qcut(df["TotalPurchaseQuantity"], q =3, labels=["Small",
```

In [117…
```
df[['OrderSize','TotalPurchaseQuantity']]
```

Out[117…

	OrderSize	TotalPurchaseQuantity
0	Large	13536
1	Large	9008
2	Large	4272
3	Large	7072
4	Medium	176
...	...	...
664	Small	32
665	Small	32
666	Medium	96
667	Small	16
668	Small	16

669 rows × 2 columns

In [118…

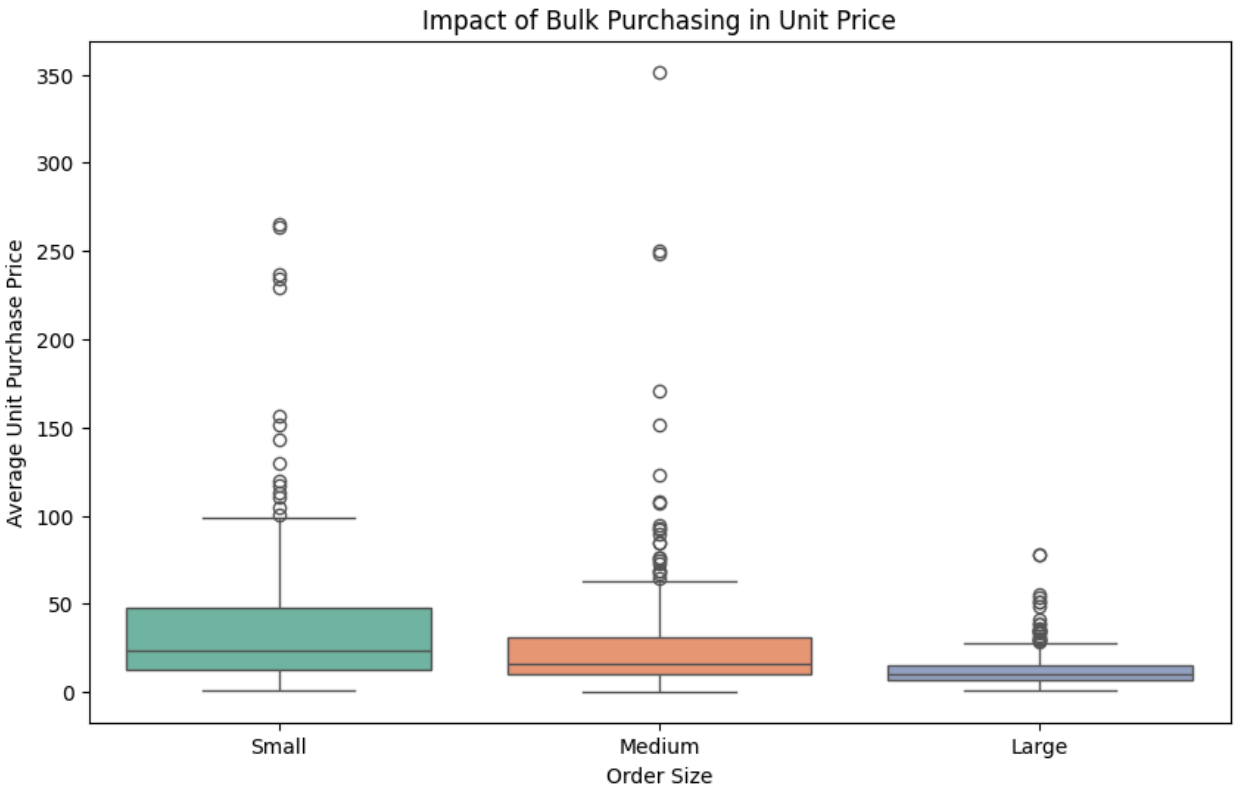
```
df.groupby('OrderSize')[['UnitPurchasePrice']].mean()
```

Out[118…

OrderSize	UnitPurchasePrice
Small	37.902423
Medium	28.452227
Large	13.319189

In [119…

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x="OrderSize", y="UnitPurchasePrice", palette="Set2")
plt.title("Impact of Bulk Purchasing in Unit Price")
plt.xlabel("Order Size")
plt.ylabel("Average Unit Purchase Price")
plt.show()
```



- Vendors buying in bulk (large ordersize) get the lowest unit price($10.78per unit), meaning higher if they can manage inventory efficiently.
- The price difference between small and large orders is substantial (~72% reduction in unit cost)
- This suggests that bulk pricing startegies successfully encourage vendors to purchase in large volumes, leading to higher overall sales despite lower per-unit revenue.

Which vendors have low inventory turnover , indicating excess stock and slow-moving producgs?

In [123…

```
df[df['StockTurnover']<1].groupby('VendorName')[['StockTurnover']].mean().sort
```

Out[123…

VendorName	StockTurnover
WINE GROUP INC	0.468750
PROXIMO SPIRITS INC.	0.679861
THE PIERPONT GROUP LLC	0.721154
BROWN-FORMAN CORP	0.735109
KLIN SPIRITS LLC	0.735577
BACARDI USA INC	0.739187
R.P.IMPORTS INC	0.743750
ATLANTIC IMPORTING COMPANY	0.743952
FORTUNE WINE BROKERS LLC	0.750000
BANFI PRODUCTS CORP	0.750000

In [124…

```
df["UnsoldInventoryValue"] = (df["TotalPurchaseQuantity"] - df["TotalSalesQuar
print('Total Unsold Captical:', format_dollars(df["UnsoldInventoryValue"].sum(
```

```
Total Unsold Captical: -936540.19
```

In [126…

```
# Aggregate capital locked per vendor
inventory_value_per_vendor = df.groupby("VendorName")["UnsoldInventoryValue"].

#Sort vendors with the highest locked capital
inventory_value_per_vendor = inventory_value_per_vendor.sort_values(by="Unsold
inventory_value_per_vendor['UnsoldInventoryValue'] = inventory_value_per_vendo
inventory_value_per_vendor.head(10)
```

Out[126…

	VendorName	UnsoldInventoryValue
0	ALISA CARR BEVERAGES	11.77K
12	DISARONNO INTERNATIONAL LLC	6.28K
52	THE PIERPONT GROUP LLC	5.84K
33	NICHE W & S	3.72K
32	MOET HENNESSY USA INC	3.59K
41	R.P.IMPORTS INC	3.20K
1	ATLANTIC IMPORTING COMPANY	2.63K
11	DIAGEO NORTH AMERICA INC	2.24K
43	Russian Standard Vodka	1.30K
39	POVERTY LANE ORCHARDS	952.24

In [ ]:

In [127…
```
top_threshold = df["TotalSalesDollars"].quantile(0.75)
low_threshold = df["TotalSalesDollars"].quantile(0.25)
```

In [130…
```
top_vendors = df[df["TotalSalesDollars"] >=top_threshold] ["ProfitMargin"].drc
low_vendors = df[df["TotalSalesDollars"] <= low_threshold]["ProfitMargin"].drc
```

In [131…
```
top_vendors
```

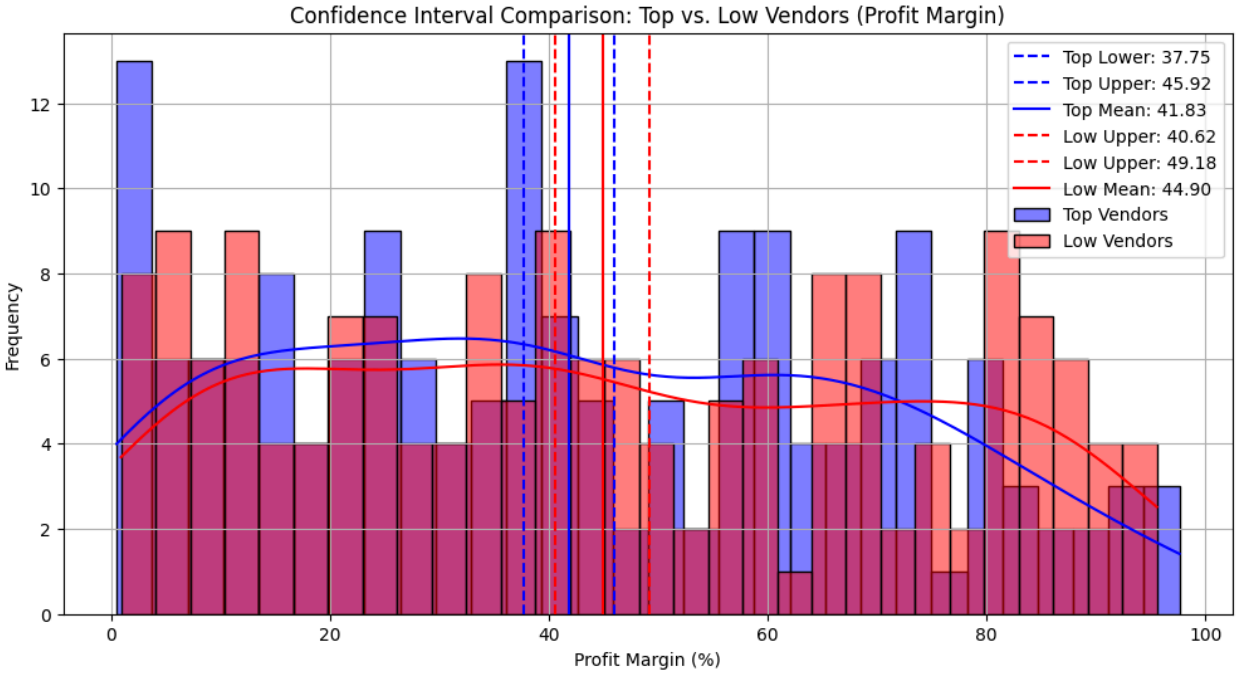
Out[131…
```
0       0.641123
1      24.171209
2      25.188353
3      13.962626
4       4.515162
          ...
401    88.495058
485    95.645673
513    93.972120
597    97.515874
600    97.617315
Name: ProfitMargin, Length: 168, dtype: float64
```

In [133…
```
def confidence_interval(data, confidence=0.95):
    mean_val = np.mean(data)
    std_err = np.std(data, ddof=1) / np.sqrt(len(data)) # standard error
    t_critical = stats.t.ppf((1+ confidence) / 2, df=len(data)-1)
    margin_of_error = t_critical * std_err
    return mean_val, mean_val-margin_of_error, mean_val + margin_of_error
```

In [137…
```
top_mean, top_lower, top_upper = confidence_interval(top_vendors)
low_mean, low_lower, low_upper = confidence_interval(low_vendors)

print(f"Top Vendors 95% CI: ({top_lower:.2f}, {top_upper:.2f}), Mean: {top_mea
print(f"Low Vendors 95% CI: ({low_lower:.2f}, {low_upper:.2f}), Mean: {low_mea

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

# Top Vendors Plot
sns.histplot(top_vendors, kde=True, color="blue", bins=30, alpha=0.5, label="T
plt.axvline(top_lower, color="blue", linestyle="--", label=f"Top Lower: {top_l
plt.axvline(top_upper, color="blue", linestyle="--", label=f"Top Upper: {top_u
plt.axvline(top_mean, color="blue", linestyle="-", label=f"Top Mean: {top_mean

# Low Vendors Plot
sns.histplot(low_vendors, kde=True, color="red", bins=30, alpha=0.5, label="Lo
plt.axvline(low_lower, color="red", linestyle="--", label=f"Low Upper: {low_lo
plt.axvline(low_upper, color="red", linestyle="--", label=f"Low Upper: {low_up
plt.axvline(low_mean, color="red", linestyle="-", label=f"Low Mean: {low_mean:

#Finalize Plot
plt.title("Confidence Interval Comparison: Top vs. Low Vendors (Profit Margin)
```

```
plt.xlabel("Profit Margin (%)")
plt.ylabel("Frequency")
plt.legend()
plt.grid(True)
plt.show()
```

```
Top Vendors 95% CI: (37.75, 45.92), Mean: 41.83
Low Vendors 95% CI: (40.62, 49.18), Mean: 44.90
```



In [ ]:

In [ ]:

In [145…

```
top_threshold = df["TotalSalesDollars"].quantile(0.75)
low_threshold = df["TotalSalesDollars"].quantile(0.25)

top_vendors = df[df["TotalSalesDollars"] >= top_threshold]["ProfitMargin"].dro
low_vendors = df[df["TotalSalesDollars"] <= low_threshold]["ProfitMargin"].dro

# Perform two-sample t-test
t_stat, p_value = ttest_ind(top_vendors, low_vendors, equal_var=False)

# Print Results
print(f"T-Statistics: {t_stat:.4f}, P-Value, {p_value:.4f}")
if p_value < 0.05:
    print("Reject H0 : There is a significant difference in profit margins bet
else:
    print("Fall to Reject H0: No significant difference in profit margins.")
```

```
T-Statistics: -1.0218, P-Value, 0.3076
Fall to Reject H0: No significant difference in profit margins.
```