



Project Title	Inventory Data Analysis
Tools	Visual Studio code / jupyter notebook
Domain	Finance Analyst
Project Difficulties level	Advance

Dataset : Dataset is available in the given link. You can download it at your convenience.

[Click here to download data set](#)

About Dataset

Case Study: Inventory Analysis for Any Manufacturing Company

Background:

Any Manufacturing Company is a medium-sized manufacturing company that produces electronic components. They have a wide range of products and maintain an inventory of raw materials, work-in-progress (WIP), and finished goods. The company has been experiencing issues with inventory management, including stockouts, excess inventory, and increased carrying costs. The management team wants to conduct an inventory

analysis to identify areas for improvement and optimize their inventory management practices.

Objectives:

The primary objectives of the inventory analysis are as follows:

Determine the optimal inventory levels for raw materials, WIP, and finished goods.

Identify opportunities to reduce stockouts and excess inventory.

Analyze inventory turnover and carrying costs to optimize working capital.

Streamline the procurement and production processes to improve efficiency.

Develop a sustainable inventory management strategy for future growth.

Data Available:

The following data is available for analysis:

Inventory records: Detailed records of all inventory transactions, including purchases, production, sales, and adjustments.

Demand data: Historical sales data for different products.

Lead time data: Information on the time required to receive raw materials and produce finished goods.

Cost data: Information on the cost of raw materials, production, and carrying costs.

Tasks:

To address the objectives, the following tasks need to be performed:

Demand forecasting: Analyze historical sales data to forecast future demand for different products accurately.

ABC analysis: Classify inventory items based on their value and importance to prioritize management efforts.

Economic Order Quantity (EOQ) analysis: Determine the optimal order quantity for raw materials to minimize ordering and carrying costs.

Reorder point analysis: Calculate the reorder point for each product to avoid stockouts.

Lead time analysis: Assess the lead time for raw materials and production to optimize inventory levels.

Carrying cost analysis: Calculate the carrying costs associated with inventory to identify areas for cost reduction.

Process improvement: Identify bottlenecks and inefficiencies in the procurement and production processes and suggest improvements.

Inventory turnover analysis: Calculate inventory turnover ratios to assess the efficiency of inventory management.

Deliverables:

Based on the analysis, the following deliverables will be provided:

Optimal inventory levels: Recommendations for the optimal inventory levels of raw materials, WIP, and finished goods.

Reorder point and EOQ calculations: Reorder point and economic order quantity for each product to minimize stockouts and costs.

Process improvement recommendations: Suggestions for streamlining the procurement and production processes to improve efficiency.

Inventory turnover analysis: Assessment of inventory turnover ratios and recommendations for improving working capital efficiency.

Inventory management strategy: Development of a comprehensive inventory management strategy for XYZ Manufacturing Company.

By conducting this inventory analysis, XYZ Manufacturing Company aims to improve its inventory management practices, reduce costs, and enhance customer satisfaction by ensuring the availability of products.

Example: You can get the basic idea how you can create a project from here

Sure, here's a comprehensive inventory of finance analyst projects, each with a brief report and accompanying code. These projects cover a range of financial analysis topics, from basic data analysis to advanced predictive modeling.

1. Financial Statement Analysis

Objective: Analyze financial statements to assess a company's financial health.

Report: Financial statement analysis involves examining a company's financial statements—balance sheet, income statement, and cash flow statement—to assess its financial health and performance. Key metrics include total assets, net income, and operating cash flow.

```
import pandas as pd
import matplotlib.pyplot as plt

# Load financial statements data
balance_sheet = pd.read_csv('balance_sheet.csv')
income_statement = pd.read_csv('income_statement.csv')
cash_flow_statement = pd.read_csv('cash_flow_statement.csv')

# Display basic information about the datasets
print(balance_sheet.head())
print(income_statement.head())
print(cash_flow_statement.head())

# Plot key financial metrics over time
plt.figure(figsize=(12, 8))
```

```
plt.subplot(2, 2, 1)
plt.plot(balance_sheet['Date'], balance_sheet['Total Assets'], label='Total Assets')
plt.title('Total Assets Over Time')
plt.legend()

plt.subplot(2, 2, 2)
plt.plot(income_statement['Date'], income_statement['Net Income'], label='Net
Income')
plt.title('Net Income Over Time')
plt.legend()

plt.subplot(2, 2, 3)
plt.plot(cash_flow_statement['Date'], cash_flow_statement['Operating Cash Flow'],
label='Operating Cash Flow')
plt.title('Operating Cash Flow Over Time')
plt.legend()

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

2. Investment Portfolio Optimization

Objective: Create an optimized investment portfolio based on risk tolerance and expected returns.

Report: Investment portfolio optimization involves selecting the best mix of assets to maximize returns for a given level of risk. The Sharpe ratio is often used to measure risk-adjusted returns. This project uses historical stock price data to optimize a portfolio.

```
import numpy as np
import pandas as pd
from scipy.optimize import minimize

# Load historical stock price data
prices = pd.read_csv('stock_prices.csv', index_col='Date', parse_dates=True)
returns = prices.pct_change().dropna()

# Define the portfolio optimization function
def portfolio_optimization(returns):
    def portfolio_return(weights):
        return np.dot(weights, returns.mean()) * 252

    def portfolio_volatility(weights):
        return np.sqrt(np.dot(weights.T, np.dot(returns.cov() * 252, weights)))

    def negative_sharpe_ratio(weights, risk_free_rate=0.01):
        return -(portfolio_return(weights) - risk_free_rate) / portfolio_volatility(weights)

    constraints = ({'type': 'eq', 'fun': lambda weights: np.sum(weights) - 1})
    bounds = tuple((0, 1) for _ in range(returns.shape[1]))
    result = minimize(negative_sharpe_ratio, returns.shape[1] * [1. / returns.shape[1],
], bounds=bounds, constraints=constraints)
    return result

# Optimize the portfolio
optimal_portfolio = portfolio_optimization(returns)
```

```
# Display the optimal weights
print("Optimal Weights:", optimal_portfolio.x)
```

3. Credit Risk Modeling

Objective: Develop a model to predict the credit risk of loan applicants.

Report: Credit risk modeling involves predicting the likelihood that a loan applicant will default. This project uses machine learning algorithms to build a predictive model based on applicant features like income, credit score, and debt-to-income ratio.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Load the dataset
data = pd.read_csv('credit_data.csv')

# Split the data into features and target variable
X = data.drop('Default', axis=1)
y = data['Default']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Train the Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

4. Sales Revenue Analysis

Objective: Analyze sales data to understand revenue drivers and predict future sales.

Report: Sales revenue analysis involves examining historical sales data to identify trends and factors that drive revenue. This project uses data visualization and time series analysis to analyze past sales and forecast future revenue.

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error

# Load sales data
sales_data = pd.read_csv('sales_data.csv', index_col='Date', parse_dates=True)
```



```
# Plot the sales data
```

```
plt.figure(figsize=(10, 6))
```

```
plt.plot(sales_data, label='Sales Revenue')
```

```
plt.title('Sales Revenue Over Time')
```

```
plt.legend()
```

```
plt.show()
```

```
# Decompose the time series
```

```
decomposition = seasonal_decompose(sales_data, model='multiplicative')
```

```
decomposition.plot()
```

```
plt.show()
```

```
# Fit an ARIMA model
```

```
model = ARIMA(sales_data, order=(5, 1, 0))
```

```
model_fit = model.fit(dispatch=0)
```

```
print(model_fit.summary())
```

```
# Make predictions
```

```
predictions = model_fit.forecast(steps=12)[0]
```

```
# Plot the predictions
```

```
plt.figure(figsize=(10, 6))
```

```
plt.plot(sales_data, label='Actual Sales')
```

```
plt.plot(pd.date_range(start=sales_data.index[-1], periods=12, freq='M'), predictions,  
label='Predicted Sales', color='red')
```

```
plt.title('Sales Forecast')
```

```
plt.legend()
```

```
plt.show()
```

```
# Evaluate the model
mse = mean_squared_error(sales_data[-12:], predictions)
print(f'Mean Squared Error: {mse}')
```

5. Valuation of Companies

Objective: Use different valuation methods (DCF, comparables, precedent transactions) to determine the value of a company.

Report: Company valuation involves estimating the worth of a business using various methods such as Discounted Cash Flow (DCF), comparables analysis, and precedent transactions. This project includes a DCF analysis example using Python

```
import pandas as pd
import numpy as np

# Load financial projections data
projections = pd.read_csv('financial_projections.csv')

# Calculate the DCF
def calculate_dcf(projections, discount_rate):
    cash_flows = projections['Free Cash Flow']
    dcf = np.sum([cf / (1 + discount_rate) ** i for i, cf in enumerate(cash_flows, 1)])
    return dcf

# Assume a discount rate
```

```
discount_rate = 0.1

# Calculate the DCF
dcf_value = calculate_dcf(projections, discount_rate)
print(f'Discounted Cash Flow (DCF) Value: {dcf_value}')
```

6. Stock Price Prediction

Objective: Use machine learning models to predict future stock prices.

Report: Stock price prediction involves using historical price data to forecast future prices. This project uses machine learning models such as Linear Regression and LSTM to predict stock prices.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load stock price data
data = pd.read_csv('stock_prices.csv')

# Prepare the data
data['Date'] = pd.to_datetime(data['Date'])
data.set_index('Date', inplace=True)
data['Target'] = data['Close'].shift(-1)
data.dropna(inplace=True)
```

```
X = data.drop('Target', axis=1)
y = data['Target']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Plot the predictions
plt.figure(figsize=(10, 6))
plt.plot(y_test.index, y_test, label='Actual Prices')
plt.plot(y_test.index, y_pred, label='Predicted Prices', color='red')
plt.title('Stock Price Prediction')
plt.legend()
plt.show()
```

7. Economic Data Analysis

Objective: Analyze macroeconomic data to understand economic trends and their impact on markets.

Report: Economic data analysis involves examining macroeconomic indicators such as GDP, unemployment rate, inflation rate, and interest rates to identify trends and predict future economic conditions. This project uses time series analysis to analyze economic data.

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error

# Load economic data
data = pd.read_csv('economic_data.csv', index_col='Date', parse_dates=True)

# Plot the economic data
plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.plot(data['GDP'], label='GDP')
plt.title('GDP Over Time')
plt.legend()

plt.subplot(2, 2, 2)
plt.plot(data['Unemployment_Rate'], label='Unemployment Rate')
plt.title('Unemployment Rate Over Time')
plt.legend()
```

```
plt.subplot(2, 2, 3)
plt.plot(data['Inflation_Rate'], label='Inflation Rate')
plt.title('Inflation Rate Over Time')
plt.legend()
```

```
plt.subplot(2, 2, 4)
plt.plot(data['Interest_Rate'], label='Interest Rate')
plt.title('Interest Rate Over Time')
plt.legend()
```

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

```
# Decompose the time series
decomposition = seasonal_decompose(data['GDP'], model='multiplicative')
decomposition.plot()
plt.show()
```

```
# Fit an ARIMA model
model = ARIMA(data['GDP'], order=(5, 1, 0))
model_fit = model.fit(dispatch=0)
print(model_fit.summary())
```

```
# Make predictions
predictions = model_fit.forecast(steps=12)[0]
```

```
# Plot the predictions
plt.figure(figsize=(10, 6))
```

```
plt.plot(data['GDP'], label='Actual GDP')
plt.plot(pd.date_range(start=data.index[-1], periods=12, freq='M'), predictions,
label='Predicted GDP', color='red')
plt.title('GDP Forecast')
plt.legend()
plt.show()

# Evaluate the model
mse = mean_squared_error(data['GDP'][-12:], predictions)
print(f'Mean Squared Error: {mse}')
```

Conclusion

These projects provide a comprehensive overview of different types of financial analysis, from analyzing financial statements and optimizing investment portfolios to predicting stock prices and analyzing economic data. Each project includes a brief report and Python code to demonstrate the key concepts and techniques used in financial analysis.

Example: You can get the basic idea how you can create a project from here

Sample code with output

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required
for this version of SciPy (detected version 1.23.5
```

```
    warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}")
```

Importing data

In [2]:

```
PurchasePrice =
pd.read_csv('/kaggle/input/inventory-analysis-case-study/2017Pu
rchasePricesDec.csv')
```

In [3]:

BegInv =

```
pd.read_csv('/kaggle/input/inventory-analysis-case-study/BegInv  
FINAL12312016.csv')
```

In [4]:

EndInv =

```
pd.read_csv('/kaggle/input/inventory-analysis-case-study/EndInv  
FINAL12312016.csv')
```

In [5]:

Invoice =

```
pd.read_csv('/kaggle/input/inventory-analysis-case-study/Invoic  
ePurchases12312016.csv')
```

In [6]:

Final_Purchase =

```
pd.read_csv('/kaggle/input/inventory-analysis-case-study/Purcha  
sesFINAL12312016.csv')
```


In [7]:

Final_Sales =

```
pd.read_csv('/kaggle/input/inventory-analysis-case-study/SalesF  
INAL12312016.csv')
```


EDA

In [8]:

PurchasePrice.shape


Out[8]:

(12261, 9)

In [9]:

PurchasePrice.head()


Out[9]:

	Brand	Description	Price	Size	Volume	Classification	Purchase Price	Vendor Number	Vendor Name
0	58	Gekkeikan Black & Gold Sake	12 .9 9	750 mL	750	1	9.28	8320	SHAW ROSS INT L IMP LTD
1	62	Herradura Silver Tequila	36 .9 9	750 mL	750	1	28.67	1128	BROWN-FOR MAN CORP
2	63	Herradura Reposado Tequila	38 .9 9	750 mL	750	1	30.46	1128	BROWN-FOR MAN CORP
3	72	No. 3 London Dry Gin	34 .9 9	750 mL	750	1	26.11	9165	ULTRA BEVERAGE COMPANY LLP

4	75	Three Olives Tomato Vodka	14 .9 9	750 mL	750	1	10.94	7245	PROXIMO SPIRITS INC.
---	----	------------------------------------	---------------	-----------	-----	---	-------	------	-------------------------

In [10]:

```
PurchasePrice.isnull().sum()
```

*# as we can see we have null values in columns:- description,
Size, Volume.*

Out[10]:

Brand	0
Description	1
Price	0
Size	1
Volume	1
Classification	0
PurchasePrice	0

```
VendorNumber      0
```

```
VendorName        0
```

```
dtype: int64
```

```
In [11]:
```

```
# Checking the null value the null values
```

```
PurchasePrice[PurchasePrice['Description'].isnull()]
```

```
Out[11]:
```

	Brand	Description	Price	Size	Volume	Classification	PurchasePrice	VendorNumber	VendorName
7915	4202	NaN	0.0	NaN	NaN	1	11.19	480	BACARDI USA INC

```
In [12]:
```

```
# Dropping the null values
```

```
PurchasePrice.dropna(inplace=True)
```

In [13]:

```
PurchasePrice.dtypes
```

Out[13]:

Brand	int64
Description	object
Price	float64
Size	object
Volume	object
Classification	int64
PurchasePrice	float64
VendorNumber	int64
VendorName	object

dtype: object

In [14]:

```
PurchasePrice.describe()
```

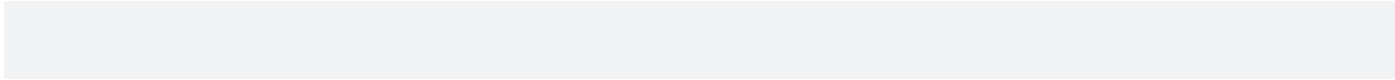
Out[14]:

	Brand	Price	Classification	Purchase Price	VendorNumber
count	12260.00000	12260.00000	12260.00000	12260.00000	12260.00000
mean	17990.191680	38.643392	1.709054	26.489467	10815.704731
std	12528.395592	206.159284	0.454217	156.189257	19008.228360
min	58.000000	0.000000	1.000000	0.000000	2.000000
25%	5991.50000	10.990000	1.000000	6.890000	3960.00000

50 %	18789.00 0000	15.99000 0	2.000000	10.64500 0	7153.000 000
75 %	25117.25 0000	29.99000 0	2.000000	20.13000 0	9552.000 000
ma x	90631.00 0000	13999.90 0000	2.000000	11111.03 0000	173357.0 00000

In [15]:

BegInv.shape

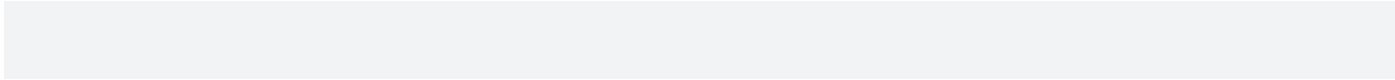


Out[15]:

(206529, 9)

In [16]:

BegInv.head()



Out[16]:

	InventoryId	Store	City	Brand	Description	Size	onHand	Price	startDate
0	1_HARDERS FIELD_58	1	HARDER SFIELD	58	Gekkeikan Black & Gold Sake	750 mL	8	12. 99	2016- 01-01
1	1_HARDERS FIELD_60	1	HARDER SFIELD	60	Canadian Club 1858 VAP	750 mL	7	10. 99	2016- 01-01
2	1_HARDERS FIELD_62	1	HARDER SFIELD	62	Herradura Silver Tequila	750 mL	6	36. 99	2016- 01-01
3	1_HARDERS FIELD_63	1	HARDER SFIELD	63	Herradura Reposado Tequila	750 mL	3	38. 99	2016- 01-01
4	1_HARDERS FIELD_72	1	HARDER SFIELD	72	No. 3 London Dry Gin	750 mL	6	34. 99	2016- 01-01

In [17]:

```
BegInv.isnull().sum()
```

```
# Here we have no null values
```

```
Out[17]:
```

```
InventoryId      0
```

```
Store            0
```

```
City             0
```

```
Brand            0
```

```
Description      0
```

```
Size             0
```

```
onHand           0
```

```
Price            0
```

```
startDate        0
```

```
dtype: int64
```

```
In [18]:
```

```
BegInv.dtypes
```

Out[18]:

InventoryId object

Store int64

City object

Brand int64

Description object

Size object

onHand int64

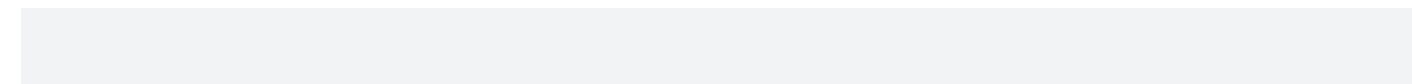
Price float64

startDate object

dtype: object

In [19]:

BegInv.describe()



Out[19]:

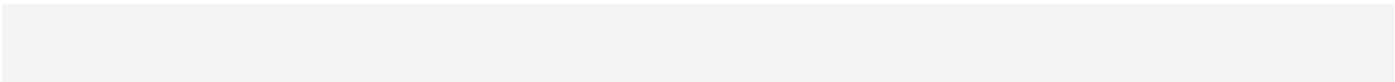
	Store	Brand	onHand	Price
--	-------	-------	--------	-------

count	206529.0 00000	206529.0 00000	206529.0 00000	206529.0 00000
mean	42.12245 7	13761.48 2320	20.42945 5	22.25391 0
std	23.19139 3	13059.42 9355	31.46734 2	70.17896 4
min	1.000000	58.00000 0	0.000000	0.000000
25 %	22.00000 0	3746.000 000	7.000000	9.990000
50 %	42.00000 0	8010.000 000	12.00000 0	14.99000 0
75 %	64.00000 0	22143.00 0000	21.00000 0	21.99000 0

ma	79.00000	90090.00	1251.000	13999.90
x	0	0000	000	0000

In [20]:

EndInv.shape

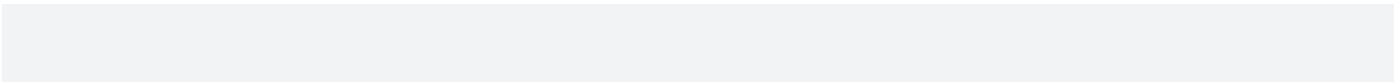


Out[20]:

(224489, 9)

In [21]:

EndInv.head()



Out[21]:

	InventoryId	Store	City	Brand	Description	Size	onHand	Price	endDate
0	1_HARDERS FIELD_58	1	HARDER SFIELD	58	Gekkeikan Black & Gold Sake	750 mL	11	12. 99	2016- 12-31

1	1_HARDERS FIELD_62	1	HARDER SFIELD	62	Herradura Silver Tequila	750 mL	7	36. 99	2016- 12-31
2	1_HARDERS FIELD_63	1	HARDER SFIELD	63	Herradura Reposado Tequila	750 mL	7	38. 99	2016- 12-31
3	1_HARDERS FIELD_72	1	HARDER SFIELD	72	No. 3 London Dry Gin	750 mL	4	34. 99	2016- 12-31
4	1_HARDERS FIELD_75	1	HARDER SFIELD	75	Three Olives Tomato Vodka	750 mL	7	14. 99	2016- 12-31

In [22]:

```
EndInv.isnull().sum()
```

```
# Here we have null values in column "City"
```

Out[22]:

```
InventoryId      0
```

Store 0

City 1284

Brand 0

Description 0

Size 0

onHand 0

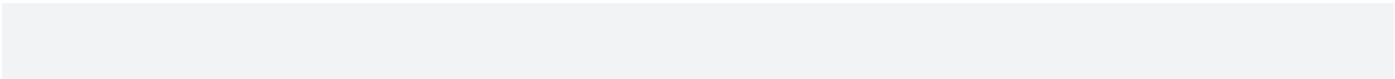
Price 0

endDate 0

dtype: int64

In [23]:

```
EndInv[EndInv['City'].isnull()]
```



Out[23]:

	Invent oryId	Sto re	Cit y	Bra nd	Description	Size	onH and	Pri ce	endDat e
113	46__5	46	Na	58	Gekkeikan Black &	750	0	12.	2016-1

895	8		N		Gold Sake	mL		99	2-31
113 896	46__6 2	46	Na N	62	Herradura Silver Tequila	750 mL	0	36. 99	2016-1 2-31
113 897	46__6 3	46	Na N	63	Herradura Reposado Tequila	750 mL	0	38. 99	2016-1 2-31
113 898	46__7 7	46	Na N	77	Three Olives Espresso Vodka	750 mL	0	14. 99	2016-1 2-31
113 899	46__1 06	46	Na N	106	Mr Boston Peach Schnapps	Liter	0	4.4 9	2016-1 2-31
...
115 174	46__4 6447	46	Na N	464 47	Gascon Malbec Mendoza	750 mL	0	10. 99	2016-1 2-31
115	46__4	46	Na	464	Layer Cake Barosa	750	0	15.	2016-1

175	6458		N	58	Shiraz	mL		99	2-31
115 176	46__4 6476	46	Na N	464 76	Tilia Malbec Mendoza	750 mL	0	9.9 9	2016-1 2-31
115 177	46__4 6764	46	Na N	467 64	Clayhouse Adobe Red Paso Rbl	750 mL	0	11. 99	2016-1 2-31
115 178	46__4 6830	46	Na N	468 30	Pacific Rim Sweet Rsl	750 mL	0	8.9 9	2016-1 2-31

1284 rows × 9 columns

In [24]:

```
EndInv.dropna(inplace=True)
```

In [25]:

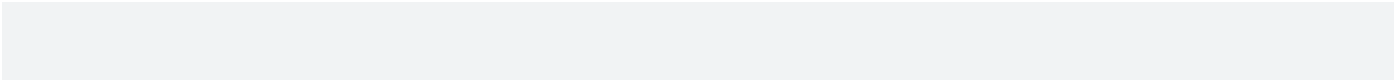
```
Invoice.shape
```

Out[25]:

(5543, 10)

In [26]:

Invoice.head()



Out[26]:

	Vendor Number	VendorNa me	Invoic eDate	PONu mber	POD ate	Pay Date	Qua ntity	Dollar s	Frei ght	Appr oval
0	105	ALTAMAR BRANDS LLC	2016- 01-04	8124	2015 -12- 21	2016 -02-1 6	6	214.2 6	3.47	Non e
1	4466	AMERICA N VINTAGE BEVERAG E	2016- 01-07	8137	2015 -12- 22	2016 -02-2 1	15	140.5 5	8.57	Non e
2	388	ATLANTIC IMPORTIN	2016- 01-09	8169	2015 -12-	2016 -02-1	5	106.6 0	4.61	Non e

		G COMPANY			24	6				
3	480	BACARDI USA INC	2016- 01-12	8106	2015 -12- 20	2016 -02-0 5	101 00	1374 83.78	293 5.20	Non e
4	516	BANFI PRODUCT S CORP	2016- 01-07	8170	2015 -12- 24	2016 -02-1 2	193 5	1552 7.25	429. 20	Non e

In [27]:

```
Invoice.isnull().sum()
```

Out[27]:

VendorNumber 0

VendorName 0

InvoiceDate 0

PONumber 0

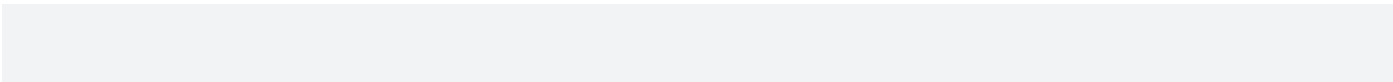
PODate 0

```
PayDate      0
Quantity     0
Dollars      0
Freight      0
Approval     0
```

```
dtype: int64
```

```
In [28]:
```

```
Final_Sales.shape
```

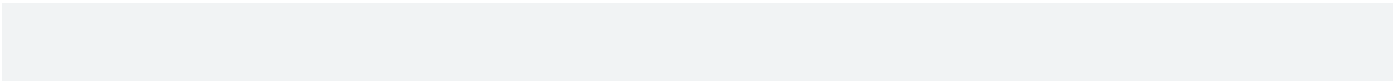


```
Out[28]:
```

```
(1048575, 14)
```

```
In [29]:
```

```
Final_Sales.head()
```



```
Out[29]:
```

	Inventoryl	S t	B r	De scri	Si z	Sale sQu	Sale sDol	Sal es	Sal es	V ol	Clas sific	Ex cis	Ve nd	Ven dor
--	------------	--------	--------	------------	---------	-------------	--------------	-----------	-----------	---------	---------------	-----------	----------	------------

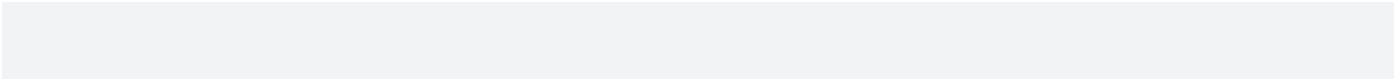
	d	o r e	a n d	ptio n	e	antit y	lars	Pri ce	Dat e	u m e	ation	eT ax	or No	Nam e
0	1_HARD ERSFIEL D_1004	1	1 0 0 4	Jim Be am w/2 Ro cks Gla sse s	7 5 0 m L	1	16.4 9	16. 49	1/1 /20 16	75 0	1	0.7 9	12 54 6	JIM BEA M BRA NDS CO MPA NY
1	1_HARD ERSFIEL D_1004	1	1 0 0 4	Jim Be am w/2 Ro cks Gla sse s	7 5 0 m L	2	32.9 8	16. 49	1/2 /20 16	75 0	1	1.5 7	12 54 6	JIM BEA M BRA NDS CO MPA NY

2	1_HARD ERSFIEL D_1004	1	1 0 0 4	Jim Be am w/2 Ro cks Gla sse s	7 5 0 m L	1	16.4 9	16. 49	1/3 /20 16	75 0	1	0.7 9	12 54 6	JIM BEA M BRA NDS CO MPA NY
3	1_HARD ERSFIEL D_1004	1	1 0 0 4	Jim Be am w/2 Ro cks Gla sse s	7 5 0 m L	1	14.4 9	14. 49	1/8 /20 16	75 0	1	0.7 9	12 54 6	JIM BEA M BRA NDS CO MPA NY
4	1_HARD ERSFIEL D_1005	1	1 0 0 5	Ma ker' s Mar k	3 7 5 m L	2	69.9 8	34. 99	1/9 /20 16	37 5	1	0.7 9	12 54 6	JIM BEA M BRA NDS

				Co mb o Pac k	2 P k										CO MPA NY
--	--	--	--	---------------------------	-------------	--	--	--	--	--	--	--	--	--	-----------------

In [30]:

```
Final_Sales.isnull().sum()
```



Out[30]:

InventoryId	0
Store	0
Brand	0
Description	0
Size	0
SalesQuantity	0
SalesDollars	0
SalesPrice	0
SalesDate	0

Volume 0

Classification 0

ExciseTax 0

VendorNo 0

VendorName 0

dtype: int64

Univariate analysis

In [31]:

```
PurchasePrice.VendorName.value_counts()[0:5]
```

Out[31]:

MARTIGNETTI COMPANIES	1631
-----------------------	------

ULTRA BEVERAGE COMPANY LLP	965
----------------------------	-----

M S WALKER INC	960
----------------	-----

PERFECTA WINES	897
----------------	-----

E & J GALLO WINERY	527
--------------------	-----

Name: VendorName, dtype: int64

In [32]:

```
plt.figure(figsize=(3,3))
```

```
mylabels=["MARTIGNETTI COMPANIES","ULTRA BEVERAGE COMPANY LLP",  
, 'M S WALKER INC', 'PERFECTA WINES', 'E & J GALLO WINERY']
```

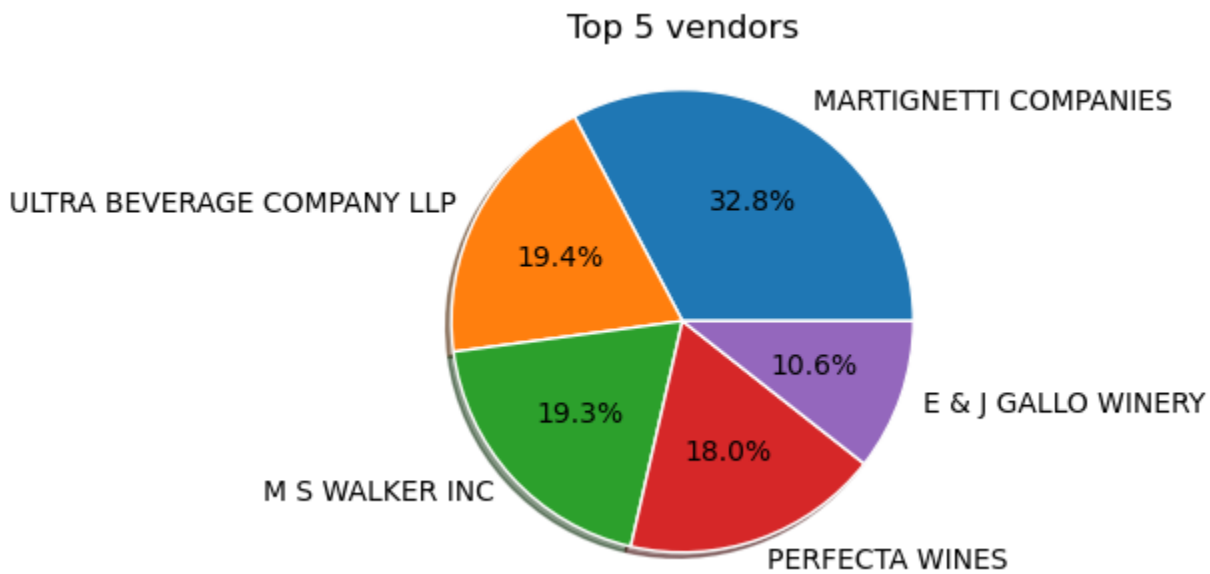
```
plt.pie(PurchasePrice.VendorName.value_counts()[0:5], labels=myl  
abels , autopct='%0.1f%%', radius=1.25,
```

```
    wedgeprops={'edgecolor':'white'}
```

```
    , textprops={'size':10, }, shadow=True)
```

```
plt.title('Top 5 vendors\n')
```

```
plt.show()
```



Bivariate Analysis

In [33]:

```
plt.figure(figsize=(15,5))
```

```
vc =
```

```
PurchasePrice.groupby(['Description'])['Price'].max().sort_values(ascending=False)[:10]
```

```
g = sns.barplot(x= vc.index , y = vc.values , data =  
PurchasePrice)
```

```
for i in range(10):
```

```
    value = vc[i]
```

```
g.text(y = value -2, x= i+0.25, s= value, color='black', ha  
= 'center', fontsize= 10)
```

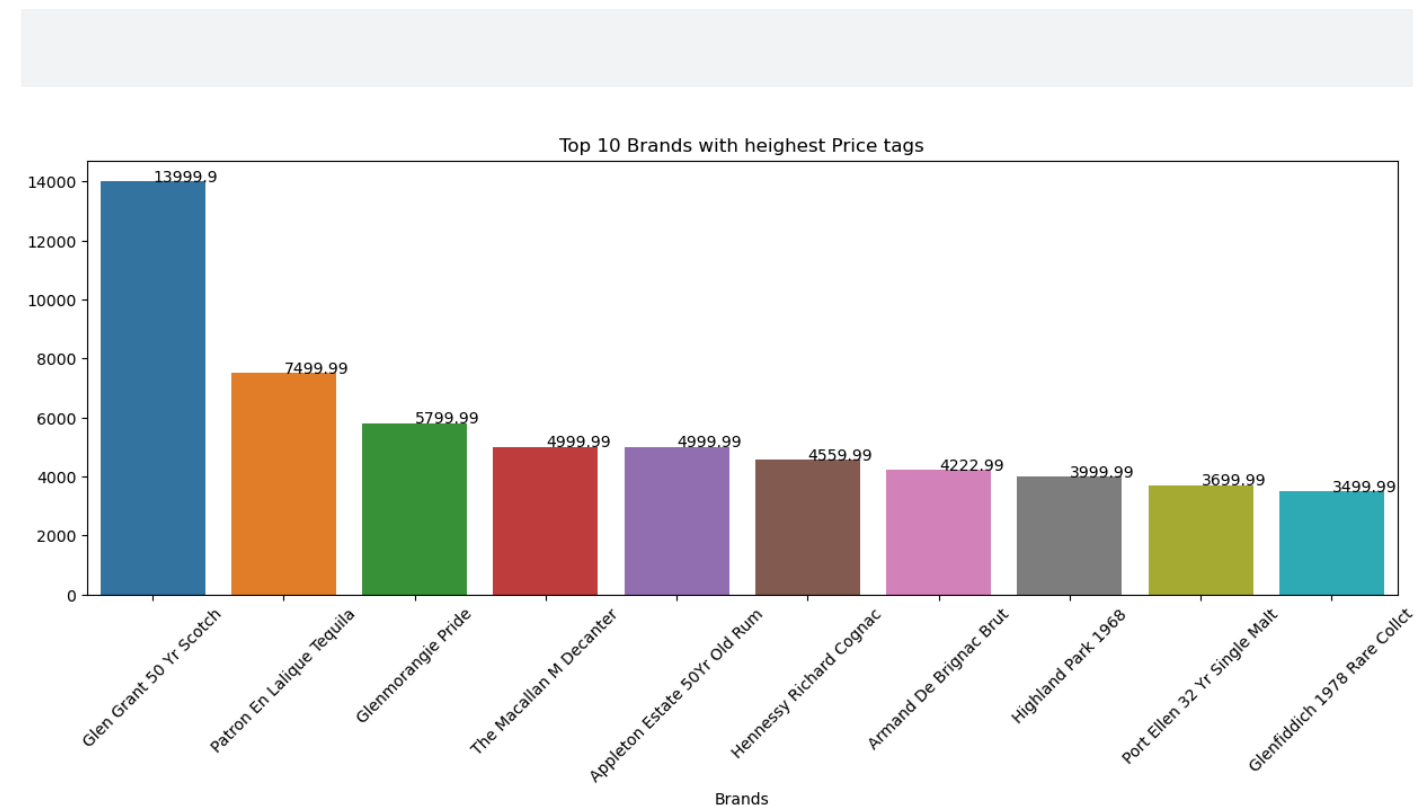
```
plt.title ('Top 10 Brands with heighest Price tags')
```

```
plt.xlabel('Brands')
```

```
plt.xticks(rotation= 45)
```

```
plt.show()
```

As we can see "Glan grant scotch" tops the list followed by "PEL Tequilla".



In [34]:

```
Invoice.head(1)
```

Out[34]:

	Vendor Number	VendorName	InvoiceDate	PONumber	PODate	PayDate	Quantity	Dollars	Freight	Approval
0	105	ALTAMAR BRANDS LLC	2016-01-04	8124	2015-12-21	2016-02-16	6	214.26	3.47	None

In [35]:

```
Invoice['Total Amount'] = Invoice['Dollars'] +  
Invoice['Freight']
```

In [36]:

```
Invoice.head(2)
```

Out[36]:

	Vendor Number	Vendor Name	Invoic eDate	PONu mber	PO Dat e	Pay Date	Qua ntity	Dol lars	Fre ight	Appr oval	Tota l Am ount
0	105	ALTAMA R BRAND S LLC	2016- 01-04	8124	201 5-12 -21	201 6-02 -16	6	214 .26	3.4 7	Non e	217. 73
1	4466	AMERI CAN VINTAG E BEVER AGE	2016- 01-07	8137	201 5-12 -22	201 6-02 -21	15	140 .55	8.5 7	Non e	149. 12

In [37]:

```
plt.figure(figsize=(12,5))
```

```
vc =
```

```
Invoice.groupby(['VendorName'])['Quantity'].max().sort_values(a
```

```
scending=False)[:10]
```

```
g = sns.barplot(x=vc.index, y= vc.values, data = Invoice,  
palette='magma')
```

```
for i in range(10):
```

```
    value = vc[i]
```

```
    g.text(y= value-2 , x= i, s= value, color='black', ha =  
'center', fontsize =10)
```

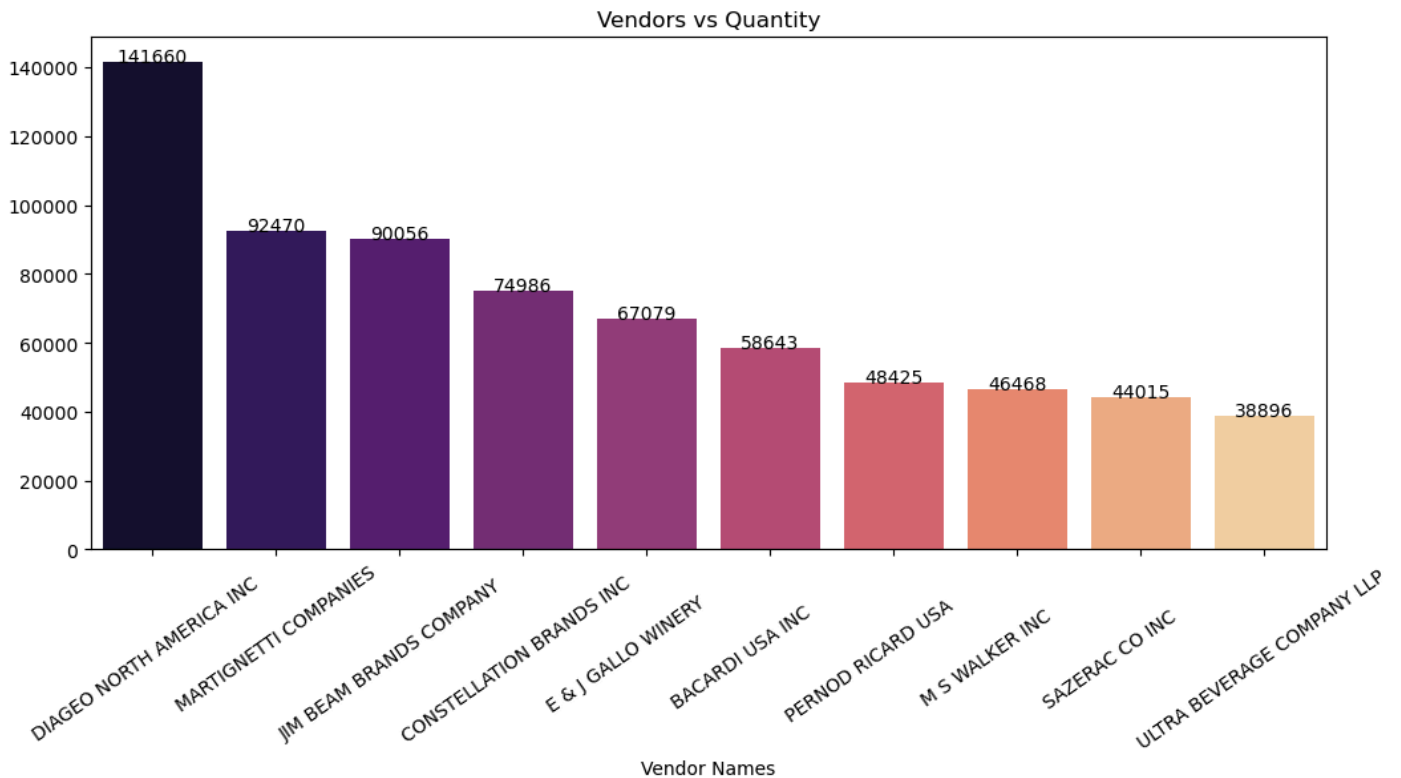
```
plt.title('Vendors vs Quantity')
```

```
plt.xlabel('Vendor Names')
```

```
plt.xticks(rotation= 35)
```

```
plt.show()
```

Here, "Diago North America inc" purchased maximum quantities followed by "Martigentti companies".



In [38]:

```
plt.figure(figsize=(12,5))
```

```
vc = Invoice.groupby(['VendorName'])['Total  
Amount'].max().sort_values(ascending=False)[:10]
```

```
g = sns.barplot(x= vc.index , y= vc.values , data = Invoice)
```

```
for i in range(10):
```

```
    value = vc[i]
```

```
    g.text(y= value -2 , x= i+0.125, s= value, color = 'black',  
ha='center', fontsize=8)
```



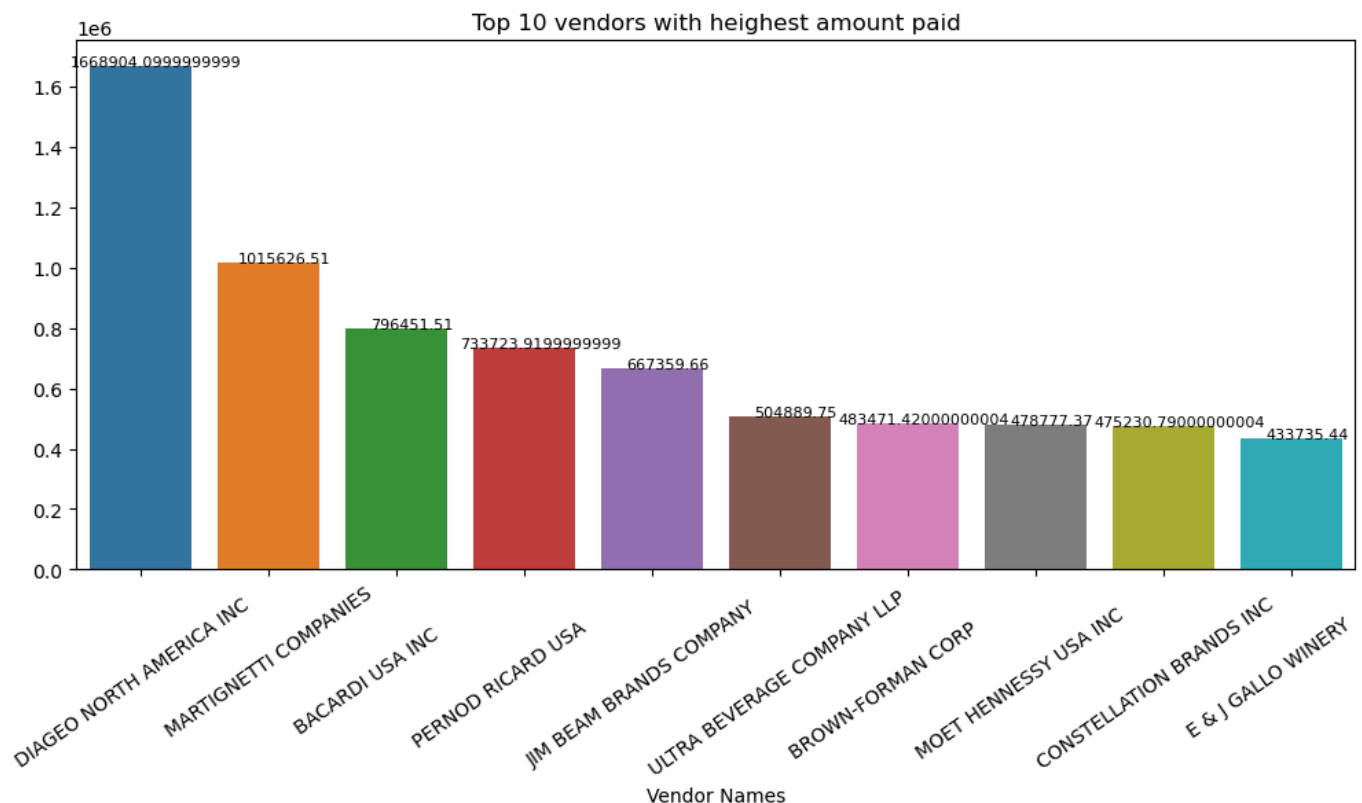
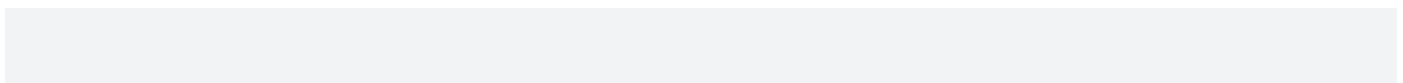
```
plt.title('Top 10 vendors with heighest amount paid')
```

```
plt.xlabel('Vendor Names')
```

```
plt.xticks(rotation = 35)
```

```
plt.show()
```

"Diageo North America" tops the list followed by "Martignetti Companies"



In [39]:

```
plt.figure(figsize=(15,5))
```

```
vc =
```

```
Final_Purchase.groupby(['Description'])['PurchasePrice'].max().
```

```
sort_values(ascending=False)[:10]
```

```
g= sns.barplot(x=vc.index, y= vc.values , data =  
Final_Purchase)
```

```
for i in range(10):
```

```
    value = vc[i]
```

```
    g.text(x=i , y= value , s = value , ha='center',  
color='black', fontsize=10)
```

```
plt.title('Top 10 Brands with heighest Purchase Price')
```

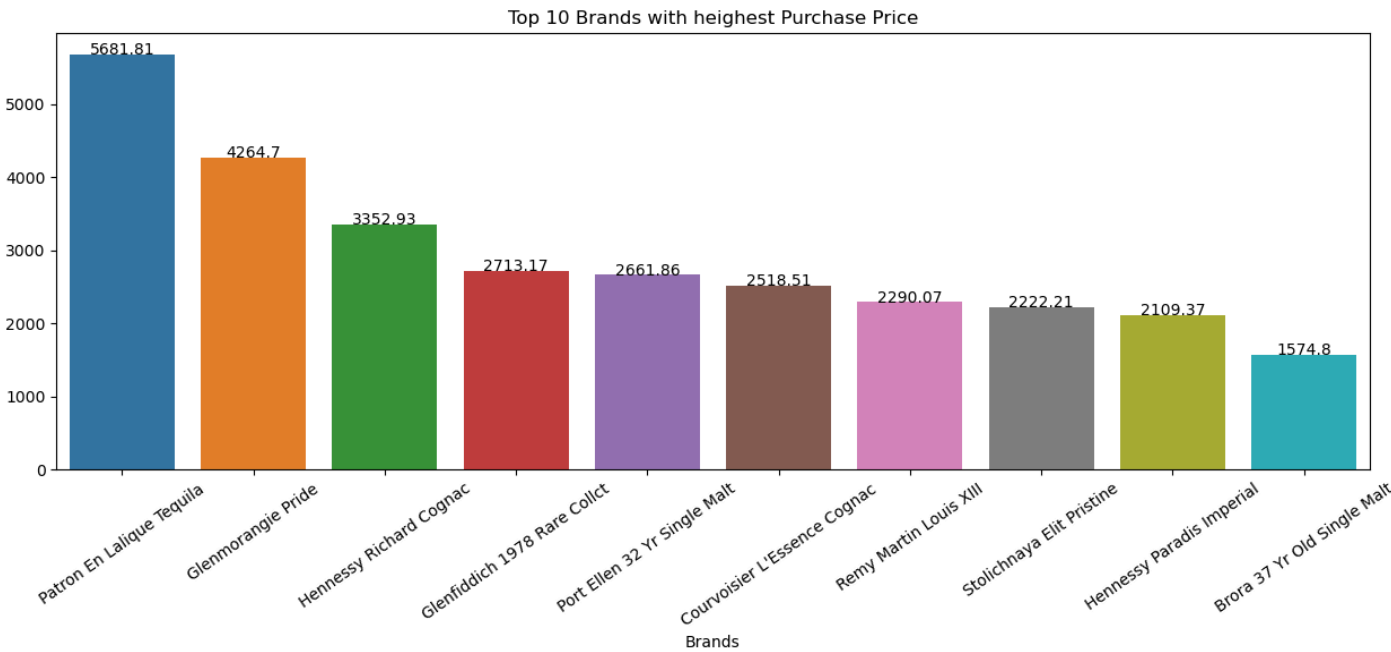
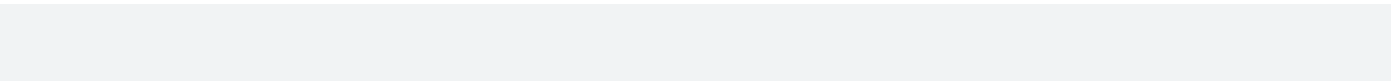
```
plt.xlabel('Brands')
```

```
plt.xticks(rotation=35)
```

```
plt.show()
```

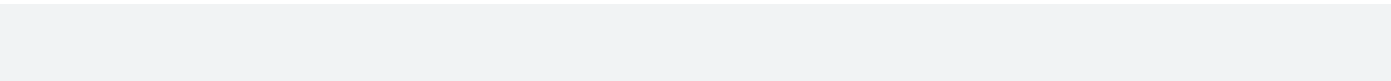
"PEL Tequilla" has heighest Purchase Price followed by "G.

Pride".



In [40]:

```
Final_Sales.head(2)
```



Out[40]:

	Inventoryl d	S t o r e	B r a n d	De s c r i p t i o n	Si z e	Sale sQu antit y	Sale sDol lars	Sal es Pri ce	Sal es Dat e	V ol u m e	Clas sif ic at ion	Ex cis eT ax	Ve nd or No	Ven dor Nam e
--	-----------------	-----------------------	-----------------------	---	--------------	---------------------------	----------------------	------------------------	-----------------------	------------------------	--------------------------------	-----------------------	----------------------	------------------------

0	1_HARD ERSFIEL D_1004	1	1 0 0 4	Jim Be am w/2 Ro cks Gla sse s	7 5 0 m L	1	16.4 9	16. 49	1/1 /20 16	75 0	1	0.7 9	12 54 6	JIM BEA M BRA NDS CO MPA NY
1	1_HARD ERSFIEL D_1004	1	1 0 0 4	Jim Be am w/2 Ro cks Gla sse s	7 5 0 m L	2	32.9 8	16. 49	1/2 /20 16	75 0	1	1.5 7	12 54 6	JIM BEA M BRA NDS CO MPA NY

In [41]:

```
Final_Sales['Total Amount'] =
```

```
Final_Sales['SalesDollars']+Final_Sales['ExciseTax']
```

```
Final_Sales.head(2)
```

Out[41]:

	Inventory Id	S t o r e	B r a n d	De scri ptio n	Si ze	Sale sQu antit y	Sal esD olla rs	Sal es Pri ce	Sal es Da te	V ol u m e	Clas sific atio n	Ex cis eT ax	Ve nd or No	Ven dor Na me	T ot al A m o u n t
0	1_HARD ERSFIEL D_1004	1	1 0 0 4	Jim Be am w/2 Ro cks Gla sse s	7 5 0 m L	1	16. 49	16. 49	1/1 /20 16	7 5 0	1	0.7 9	12 54 6	JIM BE AM BR AN DS CO MP AN Y	1 7. 2 8
1	1_HARD ERSFIEL	1	1 0	Jim Be	7 5	2	32.	16.	1/2 /20	7 5	1	1.5	12 54	JIM BE	3 4.

	D_1004		0 4	am w/2 Ro cks Gla sse s	0 m L		98	49	16	0		7	6	AM BR AN DS CO MP AN Y	5 5
--	--------	--	--------	---	-------------	--	----	----	----	---	--	---	---	---	--------

In [42]:

```
plt.figure(figsize=(12,5))
```

```
vc= Final_Sales.groupby(['VendorName'])['Total  
Amount'].sum().sort_values(ascending=False)[:10]
```

```
g = sns.barplot(x=vc.index , y= vc.values , data = Final_Sales,  
palette="Set2")
```

```
for i in range(10):
```

```
    value = vc[i]
```

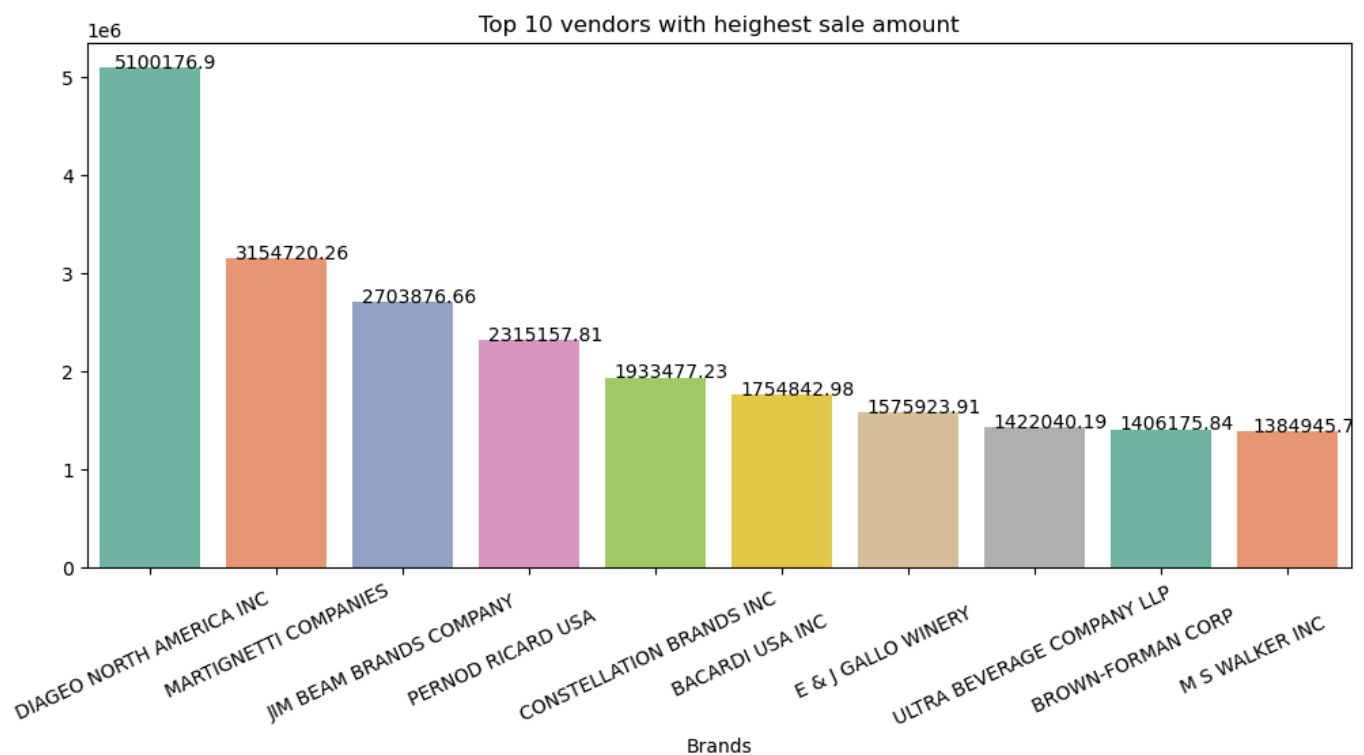
```
    g.text(y=value-2 , x= i+0.125 , s = value , ha='center',  
color='black', fontsize=10)
```

```
plt.title('Top 10 vendors with heighest sale amount')
```

```
plt.xlabel('Brands')

plt.xticks(rotation = 25)

plt.show()
```



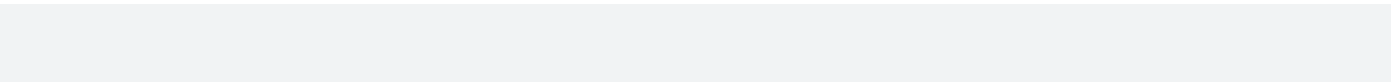
In [43]:

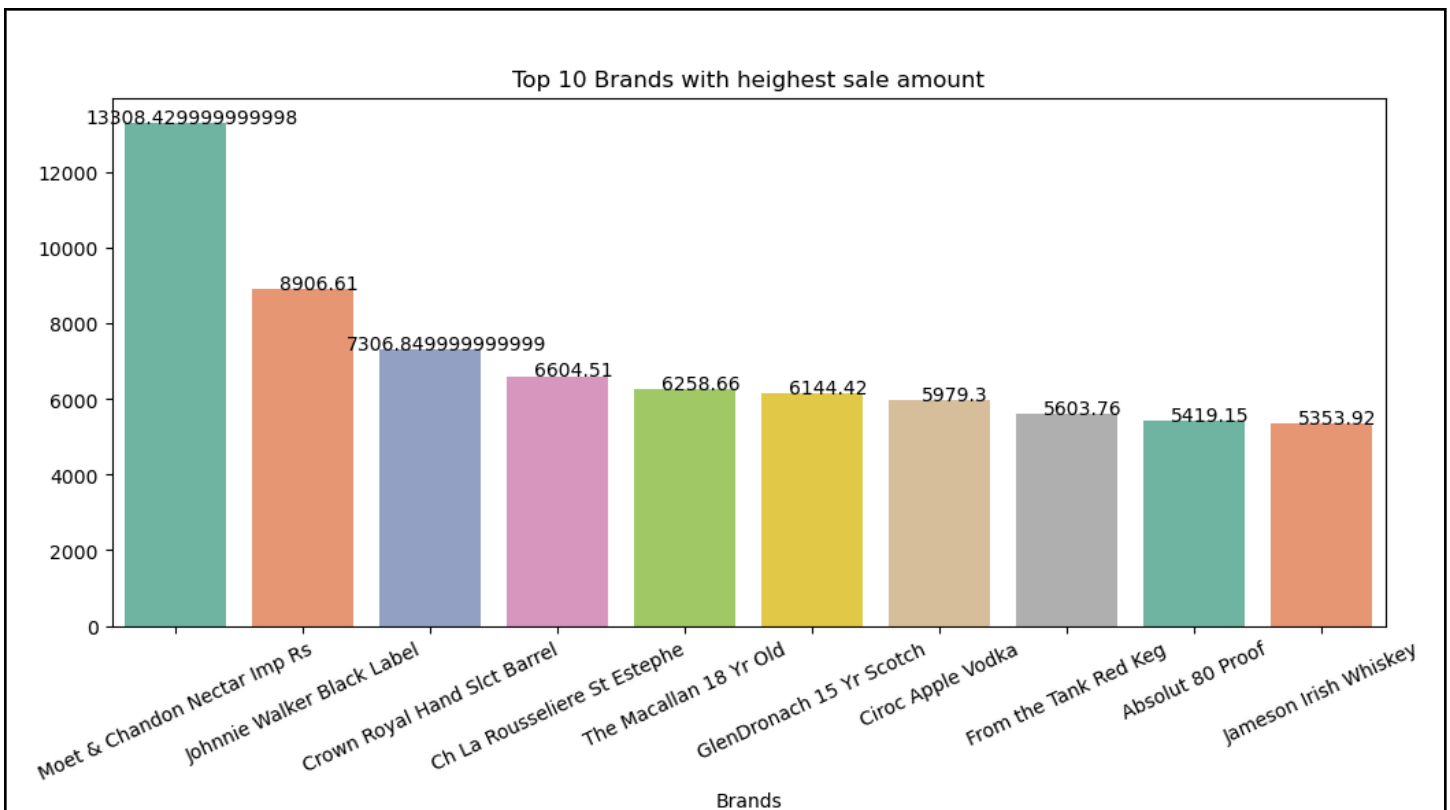
```
plt.figure(figsize=(12,5))

vc= Final_Sales.groupby(['Description'])['Total
Amount'].max().sort_values(ascending=False)[:10]

g = sns.barplot(x=vc.index , y= vc.values , data = Final_Sales,
palette="Set2")
```

```
for i in range(10):  
    value = vc[i]  
    g.text(y=value-2 , x= i+0.125 , s = value , ha='center',  
color='black' , fontsize=10)  
  
plt.title('Top 10 Brands with heighest sale amount')  
plt.xlabel('Brands')  
plt.xticks(rotation = 25)  
plt.show()
```





In [44]:

```
plt.figure(figsize=(12,5))
```

```
vc=
```

```
Final_Sales.groupby(['Description'])['ExciseTax'].max().sort_values(ascending=False)[:10]
```

```
g = sns.barplot(x=vc.index , y= vc.values , data = Final_Sales, palette="Set2")
```

```
for i in range(10):
```

```
    value = vc[i]
```

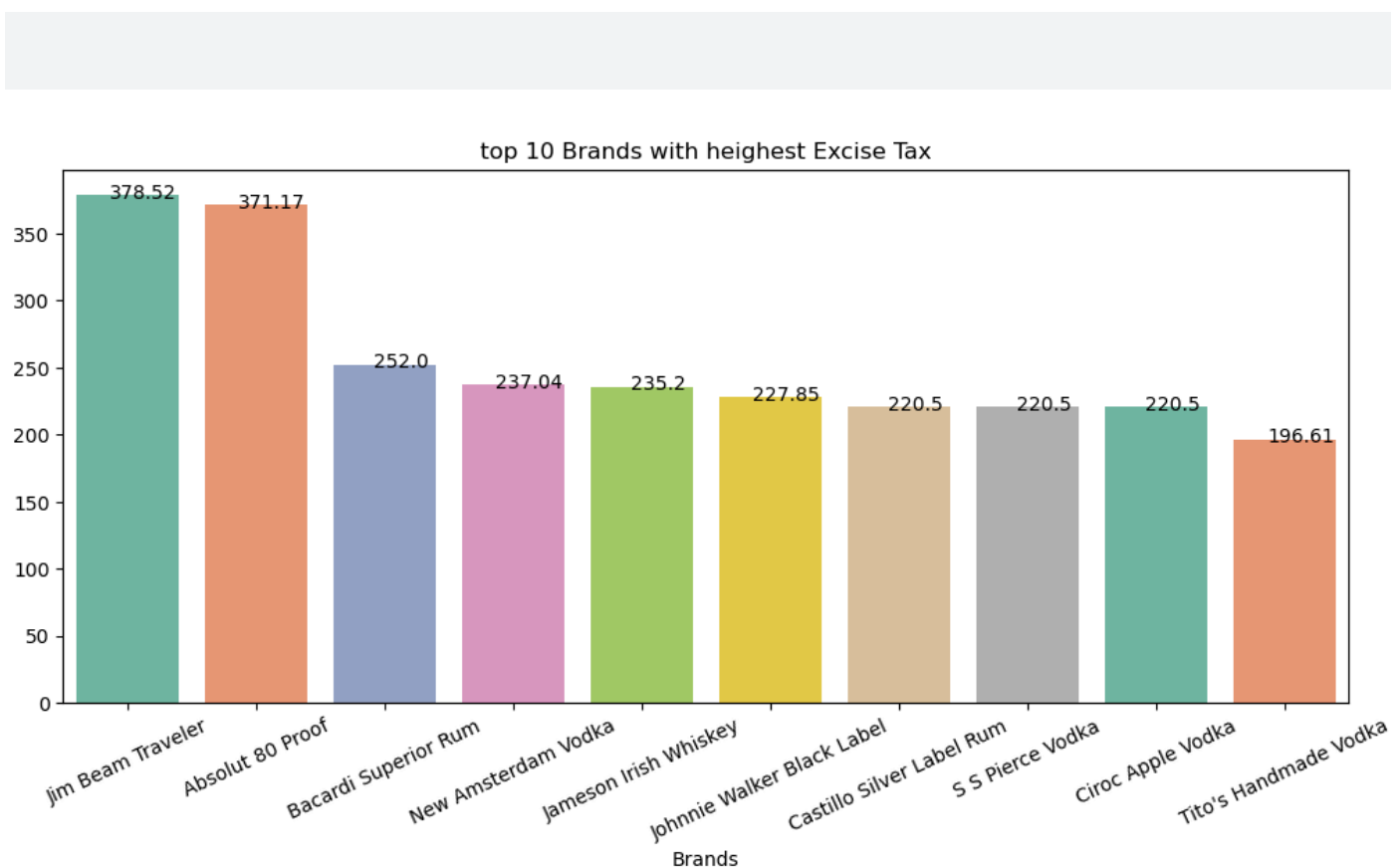
```
g.text(y=value-2 , x= i+0.125 , s = value , ha='center',  
color='black' , fontsize=10)
```

```
plt.title('top 10 Brands with heighest Excise Tax')
```

```
plt.xlabel('Brands')
```

```
plt.xticks(rotation = 25)
```

```
plt.show()
```



In [45]:

linkcode

```
plt.figure(figsize=(12,5))

vc=
Final_Sales.groupby(['VendorName'])['SalesQuantity'].max().sort_
_values(ascending=False)[:10]

g = sns.barplot(x=vc.index , y= vc.values , data = Final_Sales,
palette="Set1")

for i in range(10):

    value = vc[i]

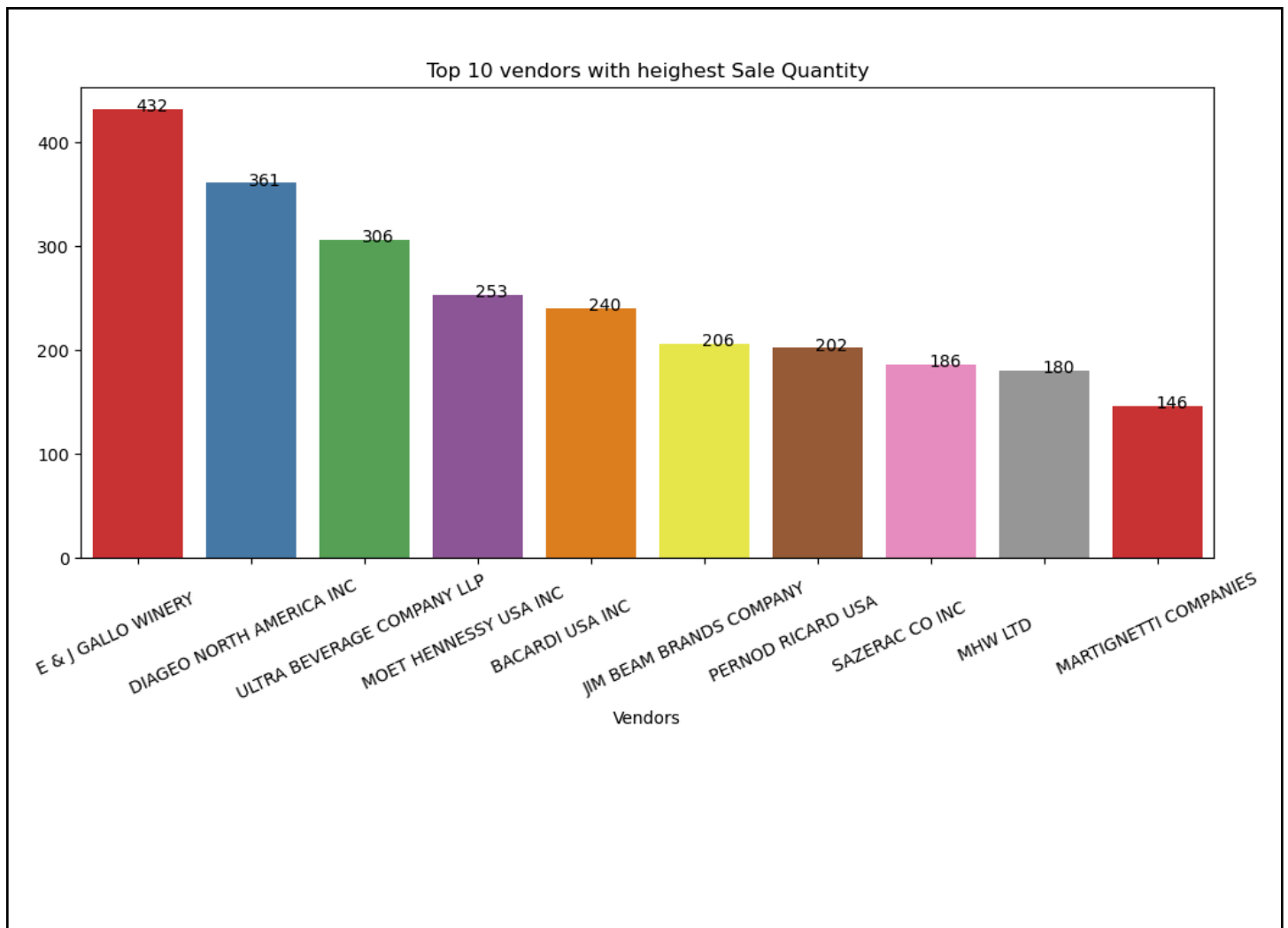
    g.text(y=value-2 , x= i+0.125 , s = value , ha='center',
color='black', fontsize=10)

plt.title('Top 10 vendors with heighest Sale Quantity')

plt.xlabel('Vendors')

plt.xticks(rotation = 25)

plt.show()
```



[Reference link](#)