VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

School of Computer Science and Engineering
Fall Semester-2024-25

Course Code : CBS3007
Course: Data Mining and Analytics

Alan Thomas

21BBS0115

Github link for the datasets and code-

https://github.com/ALANT535/DATA-MINING-RESOURCES

Aim

To better understand how to visualize the data in order to better understand the data. To know how to gather insights from any data point provided in terms of correlation between two variables or to establish a cause-effect relationship between variables

SECTION 1

Sample Input

D6 19000

Empid	Name	Designati	Salary	Experienc	Vaccinated
1	John	Manager	28000	12	yes
2	Mary	Manager	30000	12	yes
3	Michael	Superviso	23000	10	yes
4	Jennifer	Clerk	12000	2	no
5	David	superviso	19000	9	no
6	Linda	Labour	8000	1	yes
7	James	Superviso	18000	12	no
8	Susan	Clerk	11000	4	yes
9	Robert	Superviso	20000	8	no
10	Karen	Superviso	20000	8	yes
11	William	Manager	30000	9	yes
12	Patricia	Superviso	24000	8	no
13	Richard	Clerk	14000	0	yes
14	Barbara	Labour	60000	1	yes
15	Charles	Clerk	8000	2	yes
16	Jessica	Clerk	8000	3	no
17	Joseph	Clerk	8000	1	no
18	Sarah	Manager	27500	5	no
19	Thomas	Labour	30000	2	no
20	Nancy	Manager	17000	3	yes
21	Daniel	Clerk	13250	1	no
22	Lisa	Manager	24500	4	no
23	Matthew	Labour	9000	0	yes
24	Margaret	Manager	30000	7	no
25	Anthony	Superviso	8500	5	yes
26	Betty	Labour	7000	1	yes
27	Mark	Superviso	9000	6	yes
28	Dorothy	Labour	7000	2	yes
29	Steven	Labour	7000	0	yes
30	Sandra	Labour	6000	0	no

Output

```
import pandas as pd
import numpy as np
```

```
df=pd.read_csv("/content/labour.csv")
```

```
[28] # dimensions
df.shape
```

```
(30, 6)
```

```
[97] print("REG NO - 21BBS0115")
```

```
REG NO - 21BBS0115
```

```
[96] #structure
df.describe(include="all")
```

	Empid	Name	Designation	Salary	Experience	Vaccinated
count	30.000000	30	30	30.000000	30.000000	30
unique	NaN	30	5	NaN	NaN	2
top	NaN	John	Labour	NaN	NaN	yes
freq	NaN	1	8	NaN	NaN	17
mean	15.500000	NaN	NaN	17891.666667	4.600000	NaN
std	8.803408	NaN	NaN	11635.202852	3.970581	NaN
min	1.000000	NaN	NaN	6000.000000	0.000000	NaN
25%	8.250000	NaN	NaN	8125.000000	1.000000	NaN
50%	15.500000	NaN	NaN	15500.000000	3.500000	NaN
75%	22.750000	NaN	NaN	24375.000000	8.000000	NaN
max	30.000000	NaN	NaN	60000.000000	12.000000	NaN

```
[98] print("REG NO - 21BBS0115")
```

```
REG NO - 21BBS0115
```

```
# attribute name
df.columns
```

```
Index(['Empid', 'Name', 'Designation', 'Salary', 'Experience', 'Vaccinated'], dtype='object')
```

REG - 21BBS0115

```
[24] # Q2 - A
df.head(5)
```

	Empid	Name	Designation	Salary	Experience	Vaccinated
0	1	John	Manager	28000	12	yes
1	2	Mary	Manager	30000	12	yes
2	3	Michael	Supervisor	23000	10	yes
3	4	Jennifer	Clerk	12000	2	no
4	5	David	supervisor	19000	9	no

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
[99] print("REG NO - 21BBS0115")
```

```
REG NO - 21BBS0115
```

```
# Q2 - B
df.tail()
```

	Empid	Name	Designation	Salary	Experience	Vaccinated
25	26	Betty	Labour	7000	1	yes
26	27	Mark	Supervisor	9000	6	yes
27	28	Dorothy	Labour	7000	2	yes
28	29	Steven	Labour	7000	0	yes
29	30	Sandra	Labour	6000	0	no

REG - 21BBS0115

```
# Q2 - C
df.iloc[:10][["Name","Designation","Salary"]]
```

	Name	Designation	Salary
0	John	Manager	28000
1	Mary	Manager	30000
2	Michael	Supervisor	23000
3	Jennifer	Clerk	12000
4	David	supervisor	19000
5	Linda	Labour	8000
6	James	Supervisor	18000
7	Susan	Clerk	11000
8	Robert	Supervisor	20000
9	Karen	Supervisor	20000

```
[100] print("REG NO - 21BBS0115")
```

```
REG NO - 21BBS0115
```

```
# Q2 - D
df[["Name"]]
```

	Name
0	John
1	Mary
2	Michael
3	Jennifer
4	David
5	Linda
6	James
7	Susan
8	Robert
9	Karen
10	William
11	Patricia
12	Richard
13	Barbara
14	Charles
15	Jessica
16	Joseph
17	Sarah

```
[101] print("REG NO - 21BBS0115")
```

```
REG NO - 21BBS0115
```

```
# Q2 - E
df
```

	Empid	Name	Designation	Salary	Experience	Vaccinated
0	1	John	Manager	28000	12	yes
1	2	Mary	Manager	30000	12	yes
2	3	Michael	Supervisor	23000	10	yes
3	4	Jennifer	Clerk	12000	2	no
4	5	David	supervisor	19000	9	no
5	6	Linda	Labour	8000	1	yes
6	7	James	Supervisor	18000	12	no
7	8	Susan	Clerk	11000	4	yes
8	9	Robert	Supervisor	20000	8	no
9	10	Karen	Supervisor	20000	8	yes
10	11	William	Manager	30000	9	yes
11	12	Patricia	Supervisor	24000	8	no
12	13	Richard	Clerk	14000	0	yes
13	14	Barbara	Labour	60000	1	yes
14	15	Charles	Clerk	8000	2	yes
15	16	Jessica	Clerk	8000	3	no
16	17	Joseph	Clerk	8000	1	no
17	18	Sarah	Manager	27500	5	no
18	19	Thomas	Labour	30000	2	no
19	20	Nancy	Manager	17000	3	yes

```
[102] print("REG NO - 21BBS0115")
```

```
REG NO - 21BBS0115
```

```
[103] # Q3 - A
numeric_df = df.select_dtypes(include=np.number)
mean_values = numeric_df.mean()
median_values = numeric_df.median()
mode_values = numeric_df.mode()
```

Here numeric_df is the df only containing numeric values

```
[51] print("The mean values are:")
print(mean_values)
print("\nThe median values are:")
print(median_values)
```

```
The mean values are:
Empid              15.500000
Salary          17891.666667
Experience          4.600000
dtype: float64

The median values are:
Empid              15.5
Salary          15500.0
Experience          3.5
dtype: float64
```

REG NO - 21BBS0115

```
# Q3 - B
variance = numeric_df.var()
covariance = numeric_df.cov()
print("Variance of variables are -")
variance
```

Variance of variables are -

	0
Empid	7.750000e+01
Salary	1.353779e+08
Experience	1.576552e+01

dtype: float64

```
[61] print("Covariance of variables are -")
     covariance
```

Covariance of variables are -

	Empid	Salary	Experience
Empid	77.500000	-3.406466e+04	-20.965517
Salary	-34064.655172	1.353779e+08	17144.827586
Experience	-20.965517	1.714483e+04	15.765517

Next steps: Generate code with covariance | View recommended plots | New interactive sheet

```
[63] # Q3 - C
     correlation = df[['Salary', 'Experience']].corr()
     correlation
```

	Salary	Experience
Salary	1.000000	0.371112
Experience	0.371112	1.000000

Next steps: Generate code with correlation | View recommended plots | New interactive sheet

It generates a 2*2 grid. Obviously the variable will be 1.00 correlated with itself

REG NO - 21BBS0115

```
[66] # import the graph algorithms
     import seaborn as sns
     import matplotlib.pyplot as plt
```

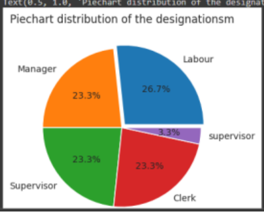
```
[81] designation_counts = df["Designation"].value_counts()
     designation_counts
```

	count
Designation	
Labour	8
Manager	7
Supervisor	7
Clerk	7
supervisor	1

dtype: int64

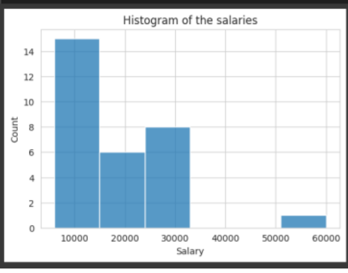
```
# Q4 - A
plt.figure(figsize=(6, 4))
sns.set_style("whitegrid")
explode = [0.05,0,0,0,0]
plt.pie(designation_counts, labels=designation_counts.index, autopct='%1.1f%%' , explode = explode)
plt.title('Piechart distribution of the designationsm')
```

Text(0.5, 1.0, 'Piechart distribution of the designationsm')



REG NO - 21BBS0115

```
# Q4 - B
plt.figure(figsize = (6,4))
sns.histplot(data = df["Salary"])
plt.title("Histogram of the salaries")
plt.show()
```



REG NO - 21BBS0115

```
# Q4 - C
plt.figure(figsize = (6,4))
sns.scatterplot(x = df["Salary"] , y = df["Experience"])
plt.title("Scatter plot of salary vs experience")
plt.show()
#
```

So we observe that there is one outlier who has very less experience but also very high salary - this is most probably an anamoly

SECTION 2

Sample Input

Month	Consumer Price Index (1982-84=1)	Motor Gasoline Price ($/gallon) Real
01-01-1999	1.65	1.79
01-02-1999	1.65	1.75
01-03-1999	1.65	1.87
01-04-1999	1.66	2.14
01-05-1999	1.66	2.14
01-06-1999	1.66	2.11
01-07-1999	1.67	2.18
01-08-1999	1.67	2.29
01-09-1999	1.68	2.35
01-10-1999	1.68	2.32
01-11-1999	1.68	2.33
01-12-1999	1.69	2.37
01-01-2000	1.69	2.39
01-02-2000	1.7	2.54
01-03-2000	1.71	2.78
01-04-2000	1.71	2.69
01-05-2000	1.71	2.73
01-06-2000	1.72	2.98
01-07-2000	1.73	2.82
01-08-2000	1.73	2.66
01-09-2000	1.74	2.8
01-10-2000	1.74	2.77
01-11-2000	1.74	2.73
01-12-2000	1.75	2.59
01-01-2001	1.76	2.59
01-02-2001	1.76	2.59
01-03-2001	1.76	2.51
01-04-2001	1.76	2.76
01-05-2001	1.77	3.01
01-06-2001	1.78	2.86

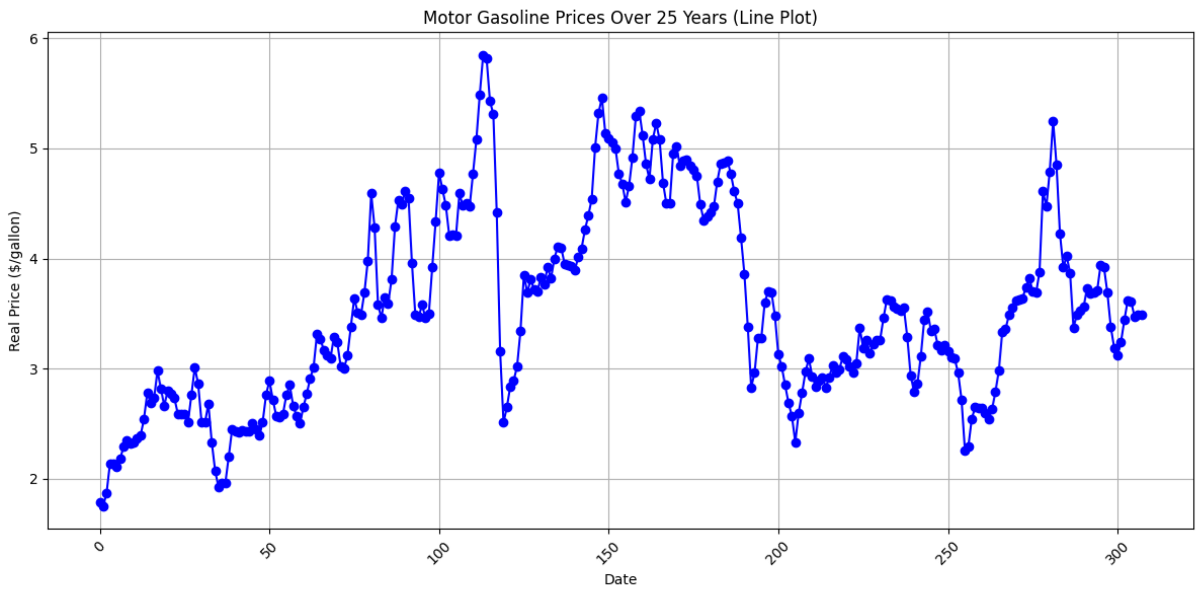
Code and Output

<u>READING DATA CODE</u>

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from scipy.stats import norm


df = pd.read_csv("/content/oil-prices.csv")

print(“REG NO – 21BBS0115”)
```

1) **Variation of Real Price Over the Dates**:

Tracks the fluctuations in real gas prices across different dates, highlighting trends, spikes, and periods of stability. This visualization helps identify patterns and significant changes in gasoline prices over time.

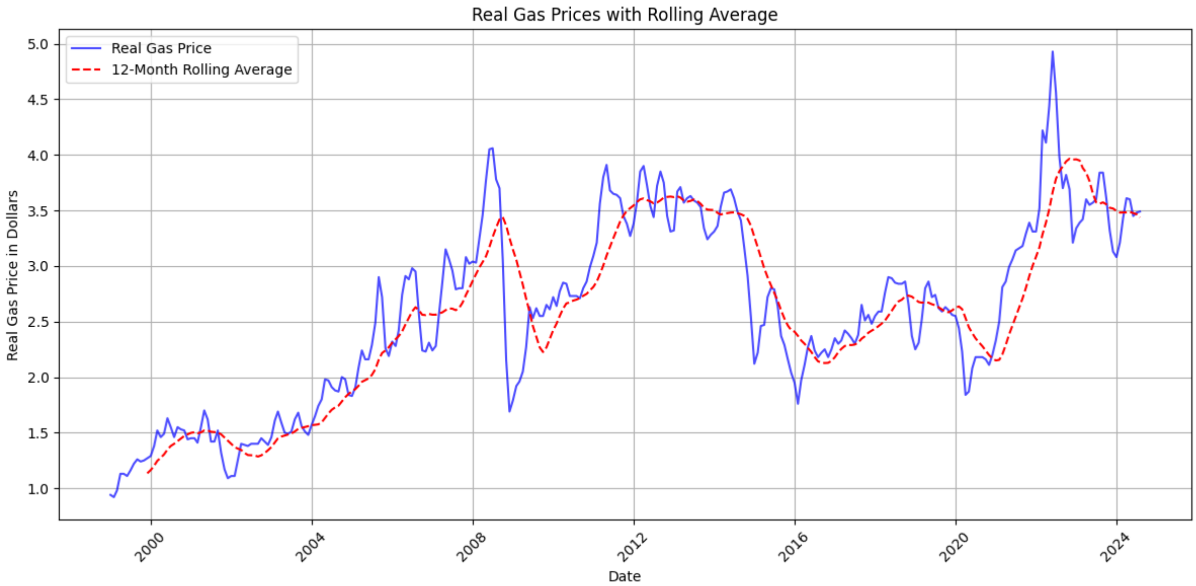


<u>Code</u>

```
plt.figure(figsize=(12, 6))
plt.plot(data.index, data['Motor Gasoline Price ($/gallon) Real'], marker='o', linestyle='-', color='b')
plt.title('Motor Gasoline Prices Over 25 Years (Line Plot)')
plt.xlabel('Date')
plt.ylabel('Real Price ($/gallon)')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

2) **Rolling Average Graph:**

Displays the real gas prices along with a smoothed rolling average over a specified period. It helps to identify long-term trends and seasonal patterns by filtering out short-term fluctuations.



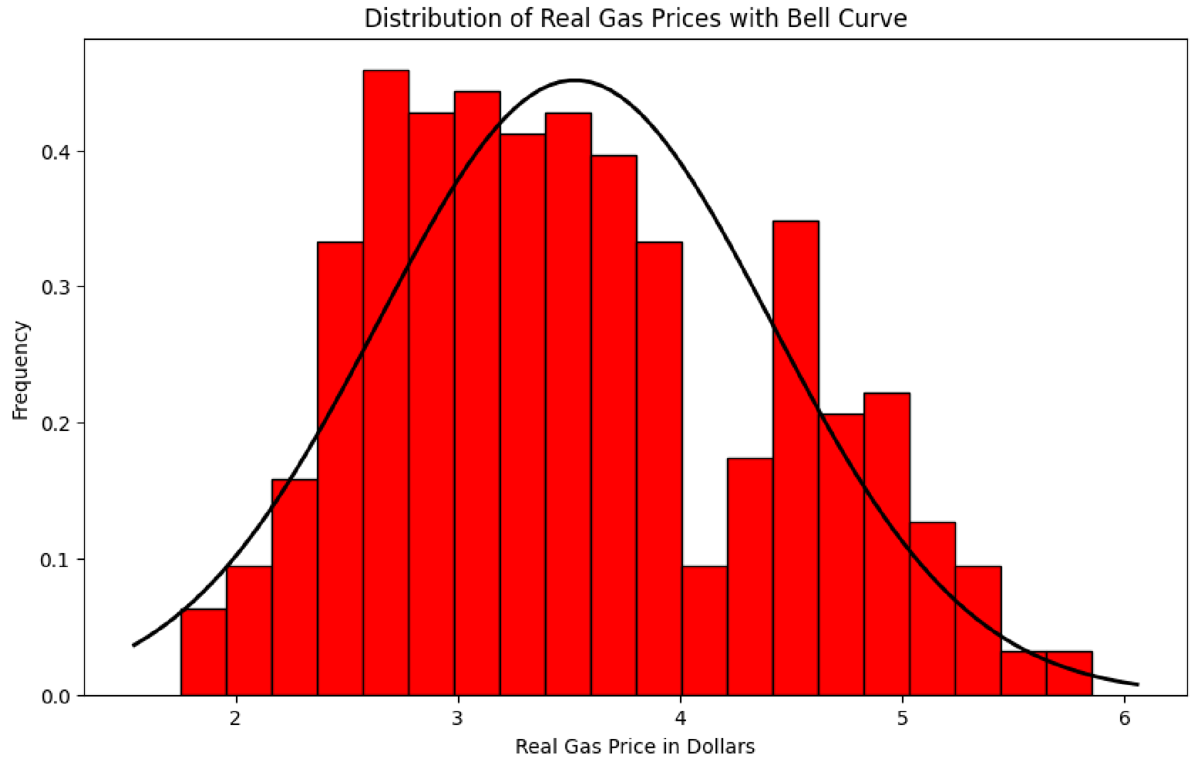
Code

```
df.columns = ['date', 'nominal_price', 'real_price', 'cpi']

df['date'] = pd.to_datetime(df['date'], format='%d-%m-%Y')

df.set_index('date', inplace=True)

window_size = 12

df['rolling_avg_real_price'] =
df['real_price'].rolling(window=window_size).mean()


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

plt.plot(df.index, df['real_price'], label='Real Gas Price', color='blue', alpha=0.7)

plt.plot(df.index, df['rolling_avg_real_price'], label=f'{window_size}-Month
Rolling Average', color='red', linestyle='--')
```

```
plt.title('Real Gas Prices with Rolling Average')
plt.xlabel('Date')
plt.ylabel('Real Gas Price in Dollars')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

3) **Distribution of Real Prices:**

Shows the frequency distribution of real gas prices, providing insight into the common price ranges and overall spread. This graph helps in understanding price variability and identifying typical price levels in the dataset.

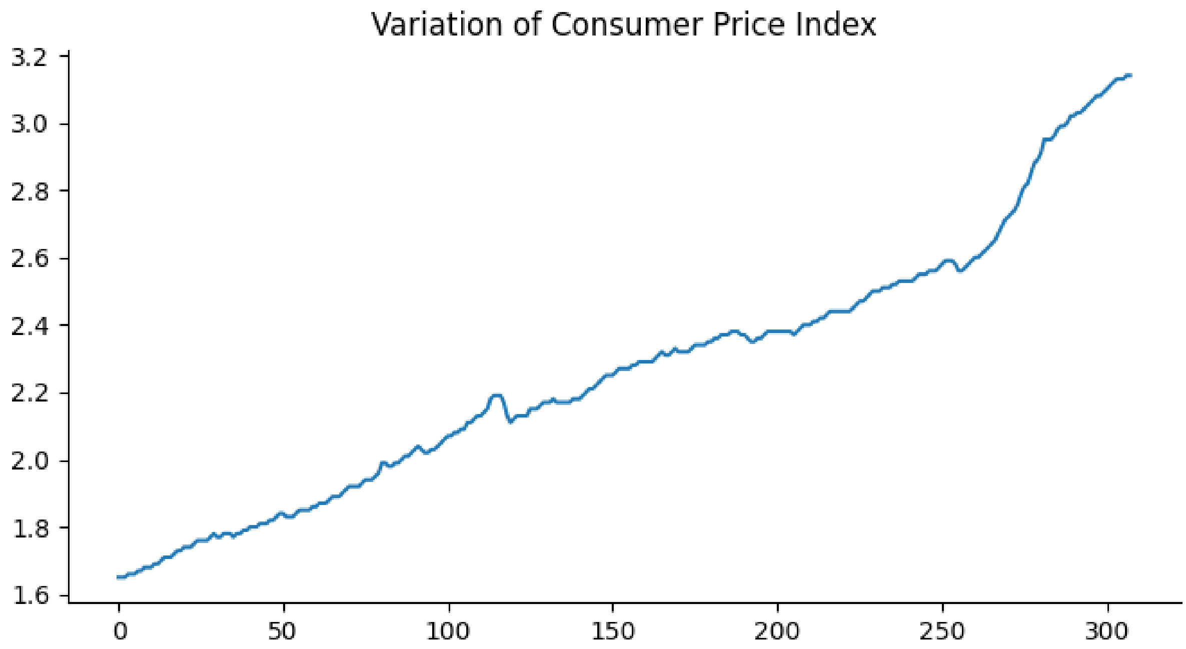


<u>Code</u>

```
df.columns = ['date','CPI','nominal price','real price']
data = df['real price'].dropna()

num_bins = 20
mean, std_dev = np.mean(data), np.std(data)

plt.figure(figsize=(10, 6))
plt.hist(data, bins=num_bins, color='red', edgecolor='black', density=True)

xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mean, std_dev)

plt.plot(x, p, 'black', linewidth=2)
plt.title('Distribution of Real Gas Prices with Bell Curve')
plt.xlabel('Real Gas Price in Dollars')
plt.ylabel('Frequency')

plt.show()
```

4) **Variation of CPI Over the Years:**

Illustrates changes in the Consumer Price Index across various years, depicting long-term inflation trends and annual shifts in price levels. This graph highlights how inflation evolves over time and impacts overall price stability.

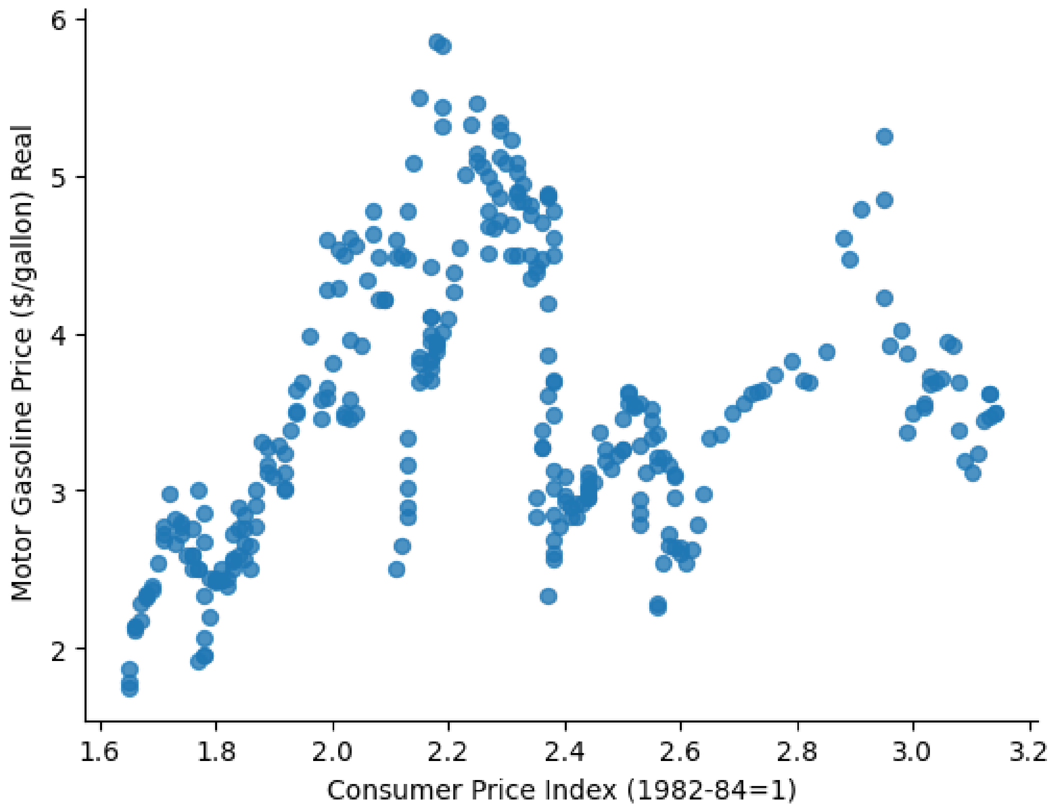


<u>Code</u>

```
df.columns = ['Month', 'Consumer Price Index (1982-84=1)',
    'Motor Gasoline Price ($/gallon) Nominal',
    ' Motor Gasoline Price ($/gallon) Real']
from matplotlib import pyplot as plt
df['Consumer Price Index (1982-84=1)'].plot(kind='line', figsize=(8, 4),
title='Variation of Consumer Price Index')
plt.gca().spines[['top', 'right']].set_visible(False)
```

5) **Correlation of CPI vs Real Price:**

Analyzes the relationship between the Consumer Price Index and real gas prices, revealing how inflationary trends affect gasoline prices over time and whether higher inflation correlates with higher real gasoline costs.



Code

```
df.columns = ['Month', 'Consumer Price Index (1982-84=1)',
    'Motor Gasoline Price ($/gallon) Nominal',
    ' Motor Gasoline Price ($/gallon) Real']

from matplotlib import pyplot as plt

df.plot(kind='scatter', x='Consumer Price Index (1982-84=1)', y=' Motor Gasoline Price ($/gallon) Real', s=32, alpha=.8)

plt.gca().spines[['top', 'right',]].set_visible(False)
```

SECTION 3

Sample Input

	A	B	C	D	E	F
1	Banana	Apple	Orange	Mango	Grapes	
2	1	1	0	0	1	
3	0	1	1	1	0	
4	1	0	1	0	1	
5	1	1	0	1	0	
6	0	1	1	1	0	
7	1	0	1	0	1	
8						
9						

<u>Code and Output</u>

<u>Code</u>

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

df = pd.read_csv('Q3\juice_stall_transactions.csv')

frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)

rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
print("Frequent Itemsets:\n", frequent_itemsets)
print("\nAssociation Rules:\n",rules)

print("\n\n21BBS0115")
```

```
PS C:\Users\LENOVO\Documents\Important_documents\VIT\Semesters\sem7\DATA MINING\DA1> python -u "c:\Users\LENOVO\Documents\Important_documents\VIT\Semester
s\sem7\DATA MINING\DA1\Q3\Q3.py"
C:\Users\LENOVO\AppData\Local\Programs\Python\Python310\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:109: DeprecationWarning: DataFrames with n
on-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
Frequent Itemsets:
     support          itemsets
0  0.666667          (Banana)
1  0.666667           (Apple)
2  0.666667          (Orange)
3  0.500000           (Mango)
4  0.500000          (Grapes)
5  0.500000  (Grapes, Banana)
6  0.500000    (Apple, Mango)

Association Rules:
   antecedents consequents  antecedent support  consequent support  support  confidence  lift  leverage  conviction  zhangs_metric
0    (Grapes)    (Banana)            0.500000            0.666667      0.5        1.00   1.5  0.166667         inf       0.666667
1    (Banana)    (Grapes)            0.666667            0.500000      0.5        0.75   1.5  0.166667         2.0       1.000000
2     (Apple)     (Mango)            0.666667            0.500000      0.5        0.75   1.5  0.166667         2.0       1.000000
3     (Mango)     (Apple)            0.500000            0.666667      0.5        1.00   1.5  0.166667         inf       0.666667


21BBS0115
PS C:\Users\LENOVO\Documents\Important_documents\VIT\Semesters\sem7\DATA MINING\DA1> 
```

SECTION 4

Sample Input

	A	B	C	D
1	Transactio	Item		
2	1	Car A		
3	1	Car B		
4	1	Service Package		
5	2	Car B		
6	2	Car C		
7	3	Car A		
8	3	Service Package		
9	4	Car A		
10	4	Car B		
11	4	Car C		
12	5	Car B		
13	5	Service Package		
14				
15				
16				

Code and Output

Code

```
import pandas as pd
from mlxtend.frequent_patterns import fpgrowth, association_rules
from mlxtend.preprocessing import TransactionEncoder

df = pd.read_csv('transactions.csv')

transactions = df.groupby('TransactionID')['Item'].apply(list).tolist()

te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df_onehot = pd.DataFrame(te_ary, columns=te.columns_)

min_support = 0.4  # 40%

frequent_itemsets = fpgrowth(df_onehot, min_support=min_support,
use_colnames=True)



print("Frequent Itemsets:")
print(frequent_itemsets)

min_confidence = 0.7  # 70%

rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=min_confidence)

print("\nAssociation Rules:")
```

```
print(rules)

print("\n\n21BBS0115")
```

```
PS C:\Users\LENOVO\Documents\Important_documents\VIT\Semesters\sem7\DATA MINING\DA1> python -u "c:\Users\LENOVO\Documents\Important_documents\VIT\Semester
s\sem7\DATA MINING\DA1\Q3.py"
Frequent Itemsets:
   support                 itemsets
0      0.8                  (Car B)
1      0.6        (Service Package)
2      0.6                  (Car A)
3      0.4                  (Car C)
4      0.4  (Car B, Service Package)
5      0.4  (Car A, Service Package)
6      0.4           (Car B, Car A)
7      0.4           (Car B, Car C)

Association Rules:
  antecedents consequents  antecedent support  consequent support  support  confidence  lift  leverage  conviction  zhangs_metric
0     (Car C)     (Car B)                 0.4                 0.8      0.4         1.0  1.25      0.08         inf       0.333333


21BBS0115
PS C:\Users\LENOVO\Documents\Important_documents\VIT\Semesters\sem7\DATA MINING\DA1> 
```

XXXXXXXXXXXXXXXXXXXXXXX