Before import csv file(i get it from kaggle) we must have a database. In mysql command line write use database_name and create a table for your csv files column the table must have name for values separately in the end i write this code load data infile 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/sales (1).csv' into table informations fields terminated by ',' lines terminated by '\n' ignore 1 lines (area_code,state,market,market_size,profit,margin,sales,COGS,total_expenses,marketing,inventory,budget_p end each cell was populated with the required data Leet's start query 1)log into to mysql from command line >> mysql -u -p 2)check local_infile variable current status >> show global variables like 'local_infile'; 3)if that is OFF,enable it >> SET GLOBAL local_infile=1; 4)quit the server >> quit 5)connect to server again >> mysql --local_infile=1 -u root -p 6)run the load sql statement.

```
CREATE SCHEMA salesinfoz;

USE salesinfoz;

create table informations( area_code INT PRIMARY KEY NOT NULL, state VARCHAR(45), market VARCHAR(45), market_size VARCHAR(45), profit DECIMAL, margin DECIMAL, sales DECIMAL, COGS DECIMAL, total_expenses DECIMAL, marketing DECIMAL, inventory DECIMAL, budget_profit DECIMAL, budget_COGS DECIMAL, budget_margin DECIMAL, budget_sales DECIMAL, productId INT, date_timez DATE, product_type VARCHAR(45), product VARCHAR(45), type VARCHAR(45) );
```

In [1]:

```
pip install ipython-sql
```

```
Requirement already satisfied: ipython-sql in c:\users\helin\anaconda3\lib
\site-packages (0.4.1)
Requirement already satisfied: six in c:\users\helin\anaconda3\lib\site-pa
ckages (from ipython-sql) (1.16.0)
Requirement already satisfied: prettytable<1 in c:\users\helin\anaconda3\l
ib\site-packages (from ipython-sql) (0.7.2)
Requirement already satisfied: ipython-genutils>=0.1.0 in c:\users\helin\a
naconda3\lib\site-packages (from ipython-sql) (0.2.0)
Requirement already satisfied: ipython>=1.0 in c:\users\helin\anaconda3\li
b\site-packages (from ipython-sql) (7.29.0)
Requirement already satisfied: sqlalchemy>=0.6.7 in c:\users\helin\anacond
a3\lib\site-packages (from ipython-sql) (1.4.22)
Requirement already satisfied: sqlparse in c:\users\helin\anaconda3\lib\si
te-packages (from ipython-sql) (0.4.3)
Requirement already satisfied: backcall in c:\users\helin\anaconda3\lib\si
te-packages (from ipython>=1.0->ipython-sql) (0.2.0)
Requirement already satisfied: decorator in c:\users\helin\anaconda3\lib\s
ite-packages (from ipython>=1.0->ipython-sql) (5.1.0)
Requirement already satisfied: pickleshare in c:\users\helin\anaconda3\lib
\site-packages (from ipython>=1.0->ipython-sql) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in c:\users\helin\anaconda3
\lib\site-packages (from ipython>=1.0->ipython-sql) (5.1.0)
Requirement already satisfied: jedi>=0.16 in c:\users\helin\anaconda3\lib
\site-packages (from ipython>=1.0->ipython-sql) (0.18.0)
Requirement already satisfied: pygments in c:\users\helin\anaconda3\lib\si
te-packages (from ipython>=1.0->ipython-sql) (2.10.0)
Requirement already satisfied: setuptools>=18.5 in c:\users\helin\anaconda
3\lib\site-packages (from ipython>=1.0->ipython-sql) (58.0.4)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.
0 in c:\users\helin\anaconda3\lib\site-packages (from ipython>=1.0->ipytho
n-sql) (3.0.20)
Requirement already satisfied: colorama in c:\users\helin\anaconda3\lib\si
te-packages (from ipython>=1.0->ipython-sql) (0.4.4)
Requirement already satisfied: matplotlib-inline in c:\users\helin\anacond
a3\lib\site-packages (from ipython>=1.0->ipython-sql) (0.1.2)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\helin\anaco
nda3\lib\site-packages (from jedi>=0.16->ipython>=1.0->ipython-sql) (0.8.
2)
Requirement already satisfied: wcwidth in c:\users\helin\anaconda3\lib\sit
e-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=1.
0->ipython-sql) (0.2.5)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\helin\anaconda
3\lib\site-packages (from sqlalchemy>=0.6.7->ipython-sql) (1.1.1)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:

```
%load_ext sql
```

In [3]:

```
%sql mysql://root:Helin2134*$@localhost/salesinfoz
```

In [4]:

```
%%sql
SELECT * FROM informations limit 10;
```

```
 * mysql://root:***@localhost/salesinfoz
10 rows affected.
```

Out[4]:

area_code	state	market	market_size	profit	margin	sales	COGS	total_expenses	m
203	Connecticut	East	Small Market	107	176	292	116	69	
203	Connecticut	East	Small Market	75	135	225	90	60	
203	Connecticut	East	Small Market	122	195	325	130	73	
203	Connecticut	East	Small Market	105	174	289	115	69	
203	Connecticut	East	Small Market	104	135	223	90	56	
203	Connecticut	East	Small Market	104	135	223	90	56	
203	Connecticut	East	Small Market	135	155	275	103	64	
203	Connecticut	East	Small Market	171	188	334	125	73	
203	Connecticut	East	Small Market	181	195	346	130	73	
203	Connecticut	East	Small Market	15	31	51	20	16	

In [5]:

```
%%sql
select * from informations
where area_code=203
limit 10;
```

```
 * mysql://root:***@localhost/salesinfoz
10 rows affected.
```

Out[5]:

area_code	state	market	market_size	profit	margin	sales	COGS	total_expenses	m
203	Connecticut	East	Small Market	107	176	292	116	69	
203	Connecticut	East	Small Market	75	135	225	90	60	
203	Connecticut	East	Small Market	122	195	325	130	73	
203	Connecticut	East	Small Market	105	174	289	115	69	
203	Connecticut	East	Small Market	104	135	223	90	56	
203	Connecticut	East	Small Market	104	135	223	90	56	
203	Connecticut	East	Small Market	135	155	275	103	64	
203	Connecticut	East	Small Market	171	188	334	125	73	
203	Connecticut	East	Small Market	181	195	346	130	73	
203	Connecticut	East	Small Market	15	31	51	20	16	

In [6]:

```
%%sql
Select DISTINCT COUNT(market)
FROM informations
GROUP BY market;
```

```
 * mysql://root:***@localhost/salesinfoz
3 rows affected.
```

Out[6]:

COUNT(market)
672
888
1344

In [7]:

```
%%sql
Select profit,margin,sales from informations limit 10;
```

```
 * mysql://root:***@localhost/salesinfoz
10 rows affected.
```

Out[7]:

profit	margin	sales
107	176	292
75	135	225
122	195	325
105	174	289
104	135	223
104	135	223
135	155	275
171	188	334
181	195	346
15	31	51

In [8]:

```
%%sql
SELECT SUM(margin+profit+sales+COGS) AS total_reality FROM informations limit 10;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[8]:

total_reality
1881064

In [9]:

```
%%sql
SELECT SUM(budget_margin+budget_profit+budget_sales+budget_COGS) AS total_budget FROM in
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[9]:

total_budget
1751080

Let's see the difference between actual demand and estimated demand in the simplest way

In [10]:

```
%%sql
SELECT (SUM(margin + profit + sales + COGS) - SUM(budget_margin + budget_profit + budget
FROM informations;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[10]:

difference
129984

we can get more detailed information with error and deviation calculations

In [11]:

```
%%sql
SELECT SUM(ABS(profit - budget_profit))/COUNT(*) AS MAD FROM informations;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[11]:

MAD
22.3194

In [12]:

```
%%sql
SELECT SUM((profit - budget_profit)*(profit - budget_profit))/COUNT(*) AS MSE FROM infor
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[12]:

MSE
1500.4508

In [13]:

```
%%sql
SELECT (SUM(ABS((profit - budget_profit) / profit ))/COUNT(*) )* 100 AS MAPE FROM inform
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[13]:

MAPE
74.55203728

calculations with based on profit of company a MAD of 22.3 could mean that the profit data is highly variable and that the company's revenue is unstable. a MAPE of 74% would generally be considered high, as it suggests that the forecasting model is not accurately predicting the profit data. This could be due to various reasons such as inaccurate or incomplete historical data, changes in market conditions, or limitations of the forecasting model. In general, a high MAPE value suggests that the profit forecasts are not reliable, and it may be necessary to improve the forecasting model or adjust the inputs to obtain more accurate results. Alternatively, it may be necessary to use other methods to forecast profit, such as trend analysis or expert opinion, if the model is unable to capture the complexity of the profit data.

general look into data

In [14]:

```
%%sql
Select distinct product_type,product, count(product) from informations group by product_
```

```
 * mysql://root:***@localhost/salesinfoz
13 rows affected.
```

Out[14]:

product_type	product	count(product)
Coffee	Columbian	480
Tea	Green Tea	288
Espresso	Caffe Mocha	480
Espresso	Decaf Espresso	408
Herbal Tea	Lemon	480
Herbal Tea	Mint	192
Tea	Darjeeling	384
Coffee	Decaf Irish Cream	384
Herbal Tea	Chamomile	384
Tea	Earl Grey	288
Espresso	Caffe Latte	216
Coffee	Amaretto	192
Espresso	Regular Espresso	72

In [15]:

```
%%sql
Select count(product_type) from informations;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[15]:

count(product_type)
4248

In [16]:

```
%%sql
Select product,sales from informations where sales=(select max(sales) from informations)
```

```
 * mysql://root:***@localhost/salesinfoz
2 rows affected.
```

Out[16]:

product	sales
Columbian	912
Columbian	912

In [17]:

```
%%sql
Select product,sales from informations where sales=(select min(sales) from informations)
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[17]:

product	sales
Green Tea	17

In [18]:

```
%%sql
Select sum(inventory) from informations;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[18]:

sum(inventory)
3183372

In [19]:

```
%%sql
Select distinct product_type,sum(inventory),sum(marketing),sum(profit) from informations
```

```
 * mysql://root:***@localhost/salesinfoz
4 rows affected.
```

Out[19]:

product_type	sum(inventory)	sum(marketing)	sum(profit)
Coffee	803954	33366	74683
Tea	760702	26738	52986
Espresso	789748	38216	68620
Herbal Tea	828968	34154	63254

In [20]:

```
%%sql
Select productId, concat(product_type," ",product," ",type) as productinfoz
from informations;
```

```
 * mysql://root:***@localhost/salesinfoz
4248 rows affected.
```

Out[20]:

productId	productinfoz
2	Coffee Columbian Regular
2	Coffee Columbian Regular
2	Coffee Columbian Regular
2	Coffee Columbian Regular
2	Coffee Columbian Regular
2	Coffee Columbian Regular
2	Coffee Columbian Regular
2	Coffee Columbian Regular
2	Coffee Columbian Regular

In [21]:

```
%%sql
select profit,product
from informations
order by profit desc limit 100;
```

```
 * mysql://root:***@localhost/salesinfoz
100 rows affected.
```

Out[21]:

profit	product
778	Columbian
777	Columbian
755	Columbian
690	Columbian
646	Regular Espresso
599	Columbian
589	Columbian
579	Columbian
572	Columbian

In [22]:

```
%%sql
select distinct product_type
FROM informations
where date_timez LIKE '2010-01-11';
```

```
 * mysql://root:***@localhost/salesinfoz
4 rows affected.
```

Out[22]:

product_type
Coffee
Tea
Espresso
Herbal Tea

is there any relation between date time and product? no.

In [23]:

```
%%sql
Create VIEW some_impo_info AS
SELECT product_type,profit,budget_profit
FROM informations
WHERE product_type='coffee' AND profit>200;
```

```
 * mysql://root:***@localhost/salesinfoz
(MySQLdb.OperationalError) (1050, "Table 'some_impo_info' already exists")
[SQL: Create VIEW some_impo_info AS
SELECT product_type,profit,budget_profit
FROM informations
WHERE product_type='coffee' AND profit>200;]
(Background on this error at: https://sqlalche.me/e/14/e3q8) (https://sqla
lche.me/e/14/e3q8))
```

i want to always see some columns with some constraints together for quick query so i use view to call them. I made it already in POPSQL so it gives this output.

In [24]:

```
%%sql
select * from some_impo_info
where profit>200;
```

```
 * mysql://root:***@localhost/salesinfoz
96 rows affected.
```

Out[24]:

product_type	profit	budget_profit
Coffee	276	400
Coffee	410	400
Coffee	364	360
Coffee	508	520
Coffee	349	310
Coffee	225	190
Coffee	292	280
Coffee	207	170
Coffee	208	230

In [25]:

```
%%sql
Select count(case when COGS>100 THEN COGS END) AS expensive_costs,
       count(COGS) AS all_costs
from informations;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[25]:

expensive_costs	all_costs
1042	4248

we see how many cost of goods sold bigger than 100

In [26]:

```
%%sql
SELECT distinct product,profit,product_type ,
sum(profit) over() as total_profit,
sum(profit) over(partition by product_type) as producttype_profit
from informations
order by product,product_type,profit desc;
```

```
 * mysql://root:***@localhost/salesinfoz
1913 rows affected.
```

Out[26]:

product	profit	product_type	total_profit	producttype_profit
Amaretto	199	Coffee	259543	74683
Amaretto	197	Coffee	259543	74683
Amaretto	190	Coffee	259543	74683
Amaretto	184	Coffee	259543	74683
Amaretto	167	Coffee	259543	74683
Amaretto	156	Coffee	259543	74683
Amaretto	153	Coffee	259543	74683
Amaretto	146	Coffee	259543	74683
Amaretto	142	Coffee	259543	74683

In [27]:

```
%%sql
CREATE INDEX idx_areacode
ON informations(area_code);
```

```
 * mysql://root:***@localhost/salesinfoz
0 rows affected.
```

Out[27]:

```
[]
```

i create index to speed up searches/queries regarding but i dont need it for this project :d

In [28]:

```
%%sql
alter table informations
drop  index idx_areacode;
```

```
 * mysql://root:***@localhost/salesinfoz
0 rows affected.
```

Out[28]:

```
[]
```

To see the distribution of the discrete values and see which variable product appears the most, we use the following.

will give us a picture of the types of products produced with the percentage

In [29]:

```
%%sql
SELECT
  product_type,
  COUNT(*) AS absolute_frequency,
  SUM(COUNT(*)) OVER (ORDER BY product_type) AS cumulative_frequency,
  100 * COUNT(*) / (SELECT COUNT(*) FROM informations) AS absolute_percentage,
  100 * SUM(COUNT(*)) OVER (ORDER BY product_type) / (SELECT COUNT(*) FROM informations)
  CONCAT(product_type, ': ', REPEAT('*', 100 * COUNT(*) / (SELECT COUNT(*) FROM informat
FROM informations
GROUP BY product_type
ORDER BY product_type;
```

```
 * mysql://root:***@localhost/salesinfoz
4 rows affected.
```

Out[29]:

product_type	absolute_frequency	cumulative_frequency	absolute_percentage	cumulative_per
Coffee	1056	1056	24.8588	
Espresso	1176	2232	27.6836	
Herbal Tea	1056	3288	24.8588	
Tea	960	4248	22.5989	

As you know, inventory cost is another issue for production, so let's look at the inventory price distribution of product types.

In [30]:

```
%%sql
SELECT inventory, COUNT(inventory) as mode
FROM informations
GROUP BY inventory
ORDER BY mode DESC
LIMIT 1;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[30]:

inventory	mode
777	54

In [31]:

```
%%sql
SELECT AVG(inventory) as median
FROM (
    SELECT @rownum:=@rownum+1 as `row_number`, t.inventory
    FROM informations t, (SELECT @rownum:=0) r
    ORDER BY t.inventory
) as tr
WHERE tr.row_number IN (FLOOR((@rownum+1)/2), FLOOR((@rownum+2)/2));
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[31]:

median
619.0000

In [32]:

```
%%sql
SELECT AVG(inventory) as mean
FROM informations;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[32]:

mean
749.3814

As a result, if we plot this mode-median-mean histogram we will see a right-skewed histogram, and sometimes the outlier histogram can be made right-skewed, so let's see if we have outliers. we use IQR for it

In [33]:

```
%%sql
SELECT
    product_type,
    inventory,
    NTILE(4) OVER (ORDER BY inventory) AS inventory_quartile
FROM informations;
```

```
 * mysql://root:***@localhost/salesinfoz
4248 rows affected.
```

Out[33]:

product_type	inventory	inventory_quartile
Coffee	-3534	1
Coffee	-3534	1
Coffee	-3287	1
Coffee	-3287	1
Coffee	-3004	1
Coffee	-3004	1
Coffee	-2572	1
Coffee	-2572	1
Espresso	-2248	1

In [34]:

```
%%sql
SELECT
    inventory_quartile,
    MAX(inventory) AS quartile_break
FROM(
    SELECT
        product_type,
        inventory,
        NTILE(4) OVER (ORDER BY inventory) AS inventory_quartile
    FROM informations) AS quartiles
WHERE inventory_quartile IN (1, 3)
GROUP BY inventory_quartile;
```

```
 * mysql://root:***@localhost/salesinfoz
2 rows affected.
```

Out[34]:

inventory_quartile	quartile_break
1	432
3	910

IQR=910-432=478 478*1,5=717 717-432=285 910+717=1627

In [35]:

```
%%sql
SELECT product_type,
inventory
FROM informations
WHERE inventory < 285 OR inventory > 1627;
```

```
 * mysql://root:***@localhost/salesinfoz
658 rows affected.
```

Out[35]:

product_type	inventory
Coffee	1832
Coffee	2947
Coffee	2108
Coffee	2108
Espresso	1744
Espresso	1744
Espresso	3641
Espresso	2615
Herbal Tea	3948

All data 4248 and we have 658 outliers, it shows %15 of data is include outlier %15 is low rate i dont think it has a impact on data but let's look our data without outliers for guarantee

In [36]:

```
%%sql
SELECT inventory, COUNT(inventory) as mode
FROM informations
WHERE inventory > 285 OR inventory < 1627
GROUP BY inventory
ORDER BY mode DESC
LIMIT 1;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[36]:

inventory	mode
777	54

In [37]:

```
%%sql
SELECT AVG(inventory) as median
FROM (
    SELECT @rownum:=@rownum+1 as `row_number`, t.inventory
    FROM informations t, (SELECT @rownum:=0) r
    ORDER BY t.inventory
) as tr
WHERE tr.row_number IN (FLOOR((@rownum+1)/2), FLOOR((@rownum+2)/2)) AND inventory > 285
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[37]:

median
633.8283

In [38]:

```
%%sql
SELECT AVG(inventory) as mean
FROM informations
WHERE inventory > 285 OR inventory < 1627;
```

```
 * mysql://root:***@localhost/salesinfoz
1 rows affected.
```

Out[38]:

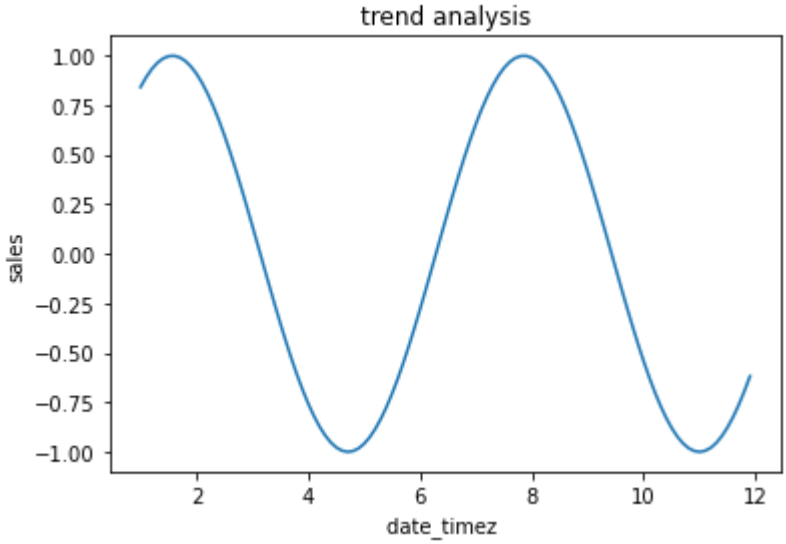
mean
749.3814

As you can see in the result it's not about outliers. However, we can interpret that the frequency of the data is not symmetrical, but instead distributed like a tail with high values.

```
Lets make some visualizations to see values clearly
```

In [39]:

```
import matplotlib.pyplot as plt
import numpy as np

# start in jan as 1
x = np.arange(1, 12, 0.1)
y = np.sin(x)


plt.plot(x, y)


plt.title("trend analysis")
plt.ylabel("sales")
plt.xlabel("date_timez ")

# Show the plot
plt.show()
```



In [40]:

```
%%sql
SELECT DISTINCT product_type ,COUNT(product_type) AS avg_count
FROM informations
GROUP BY product_type;
```

```
 * mysql://root:***@localhost/salesinfoz
4 rows affected.
```

Out[40]:

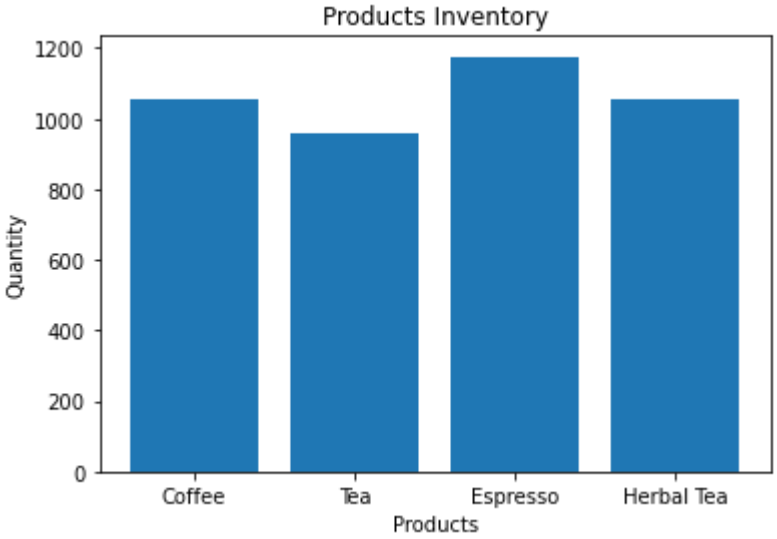
product_type	avg_count
Coffee	1056
Tea	960
Espresso	1176
Herbal Tea	1056

In [41]:

```
import matplotlib.pyplot as plt

labels = ['Coffee', 'Tea', 'Espresso', 'Herbal Tea']
values = [1056, 960, 1176, 1056]

fig, ax = plt.subplots()

ax.bar(labels, values)

ax.set_xlabel('Products')
ax.set_ylabel('Quantity')
ax.set_title('Products Inventory')
```

Out[41]:

```
Text(0.5, 1.0, 'Products Inventory')
```



In [56]:

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

# Load some sample data
df = pd.read_csv('C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/sales (1).csv')

# Create a scatter plot of two columns
plt.scatter(df['Budget Profit'], df['Profit'])

# Add some labels and a title
plt.xlabel('BUDGET PROFIT')
plt.ylabel('PROFIT')
plt.title('Scatter Plot')

# Display the plot
plt.show()
```