MIT | Academy of Engineering

EDS ACTIVITY NO. 1

NAME : ANIKET ANIL BHUJAD

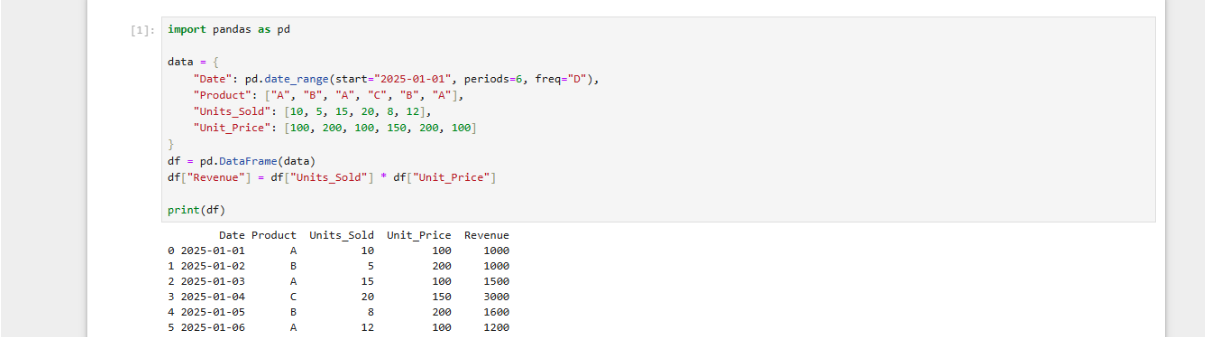
DIV : CS6

PRN : 202401080037

ROLL NO. CS6-87

BATCH : C64

- For pandas : -
  - Dataset:



```
[1]: import pandas as pd

     data = {
         "Date": pd.date_range(start="2025-01-01", periods=6, freq="D"),
         "Product": ["A", "B", "A", "C", "B", "A"],
         "Units_Sold": [10, 5, 15, 20, 8, 12],
         "Unit_Price": [100, 200, 100, 150, 200, 100]
     }
     df = pd.DataFrame(data)
     df["Revenue"] = df["Units_Sold"] * df["Unit_Price"]

     print(df)
```

```
        Date Product  Units_Sold  Unit_Price  Revenue
0 2025-01-01       A          10         100     1000
1 2025-01-02       B           5         200     1000
2 2025-01-03       A          15         100     1500
3 2025-01-04       C          20         150     3000
4 2025-01-05       B           8         200     1600
5 2025-01-06       A          12         100     1200
```

Grains : -

1.Show the first few rows of the dataset

2.Get descriptive statistics for all columns

3.Filter rows where Units_Sold > 10

4.Add a new column 'Tax' as 10% of Revenue

5.Sort rows by Revenue in descending order

6.Find the mean of Units_Sold and Revenue columns

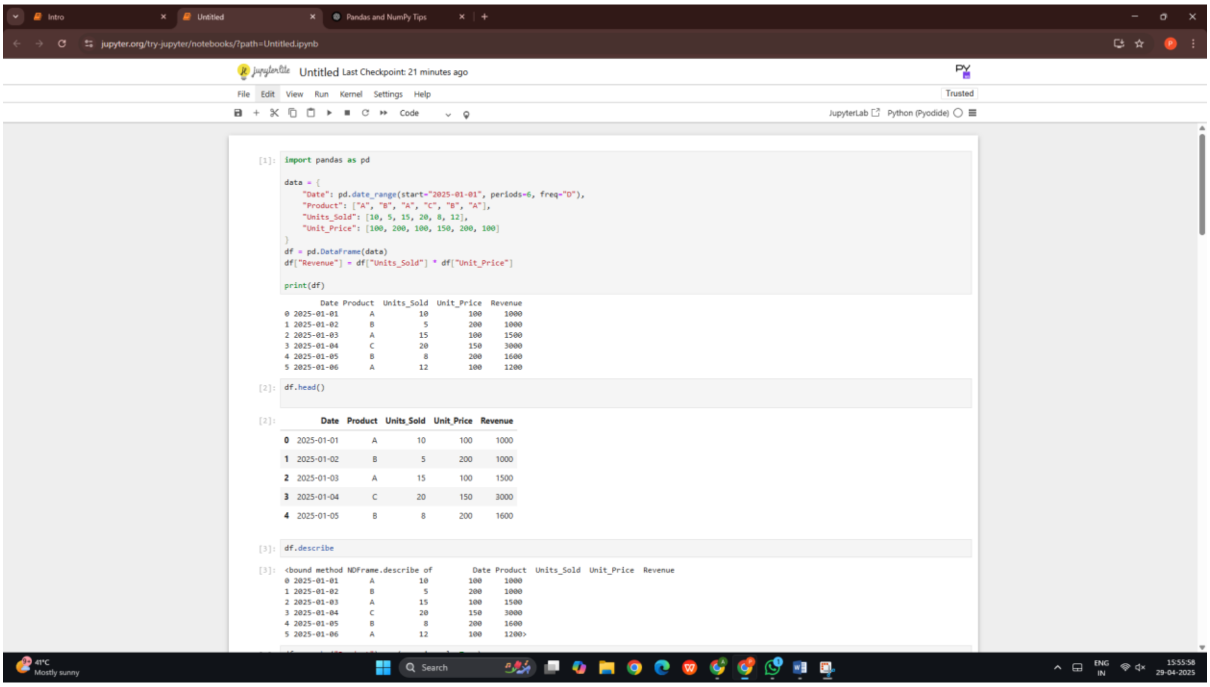
7.Filter rows where Discount is 0

8.Group by Unit_Price and get sum of values

9.Create a Discounted_Revenue column after applying discount

10.Select only the first 3 columns of the dataframe

Outputs in jupyter notebook for all grains: -



[1]:

```
import pandas as pd

data = {
    "Date": pd.date_range(start="2025-01-01", periods=6, freq="D"),
    "Product": ["A", "B", "A", "C", "B", "A"],
    "Units_Sold": [10, 5, 15, 20, 8, 12],
    "Unit_Price": [100, 200, 100, 150, 200, 100]
}
df = pd.DataFrame(data)
df["Revenue"] = df["Units_Sold"] * df["Unit_Price"]

print(df)
```

```
        Date Product  Units_Sold  Unit_Price  Revenue
0 2025-01-01       A          10         100     1000
1 2025-01-02       B           5         200     1000
2 2025-01-03       A          15         100     1500
3 2025-01-04       C          20         150     3000
4 2025-01-05       B           8         200     1600
5 2025-01-06       A          12         100     1200
```

[2]:

```
df.head()
```

[2]:

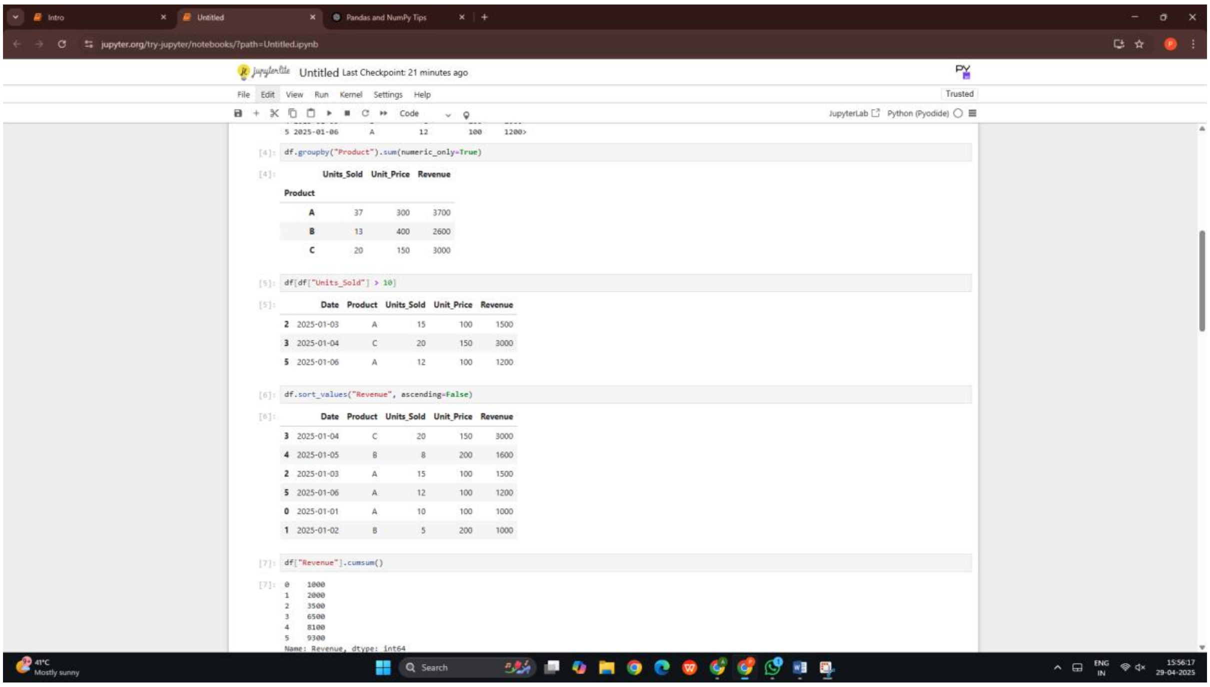
	Date	Product	Units_Sold	Unit_Price	Revenue
0	2025-01-01	A	10	100	1000
1	2025-01-02	B	5	200	1000
2	2025-01-03	A	15	100	1500
3	2025-01-04	C	20	150	3000
4	2025-01-05	B	8	200	1600

[3]:

```
df.describe
```

[3]:

```
<bound method NDFrame.describe of         Date Product  Units_Sold  Unit_Price  Revenue
0 2025-01-01       A          10         100     1000
1 2025-01-02       B           5         200     1000
2 2025-01-03       A          15         100     1500
3 2025-01-04       C          20         150     3000
4 2025-01-05       B           8         200     1600
5 2025-01-06       A          12         100     1200>
```



```
5 2025-01-06       A          12         100     1200>
```

[4]:

```
df.groupby("Product").sum(numeric_only=True)
```

[4]:

	Units_Sold	Unit_Price	Revenue
Product			
A	37	300	3700
B	13	400	2600
C	20	150	3000

[5]:

```
df[df["Units_Sold"] > 10]
```

[5]:

	Date	Product	Units_Sold	Unit_Price	Revenue
2	2025-01-03	A	15	100	1500
3	2025-01-04	C	20	150	3000
5	2025-01-06	A	12	100	1200

[6]:

```
df.sort_values("Revenue", ascending=False)
```

[6]:

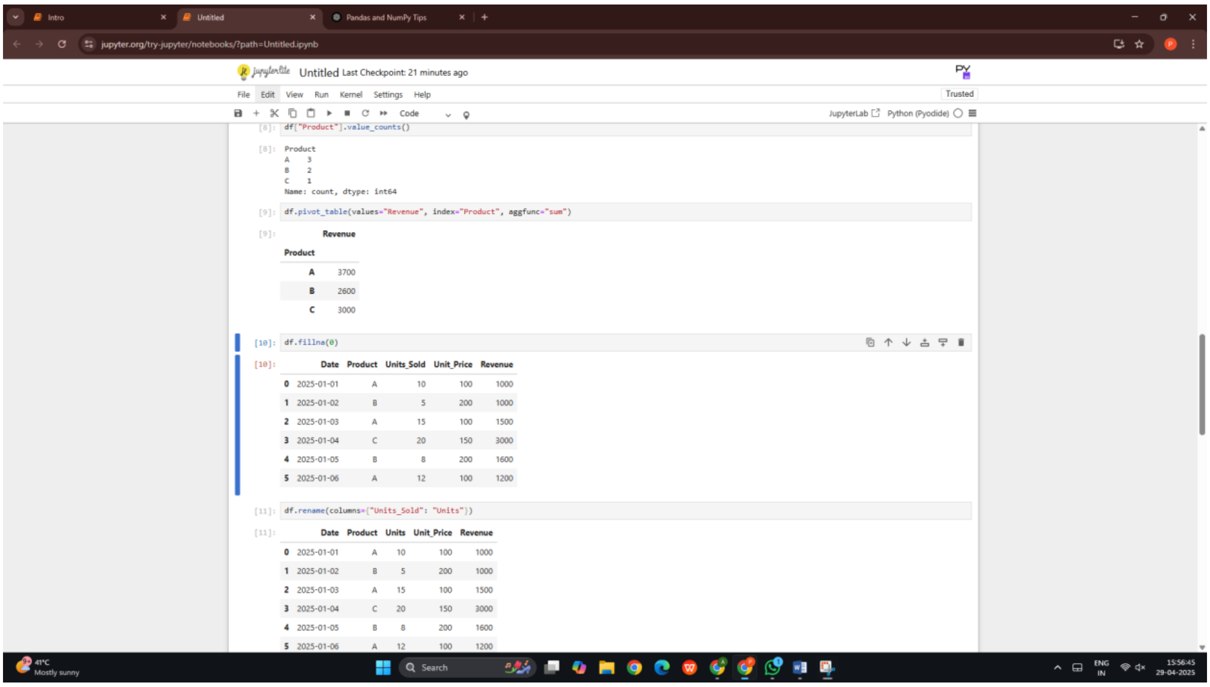
	Date	Product	Units_Sold	Unit_Price	Revenue
3	2025-01-04	C	20	150	3000
4	2025-01-05	B	8	200	1600
2	2025-01-03	A	15	100	1500
5	2025-01-06	A	12	100	1200
0	2025-01-01	A	10	100	1000
1	2025-01-02	B	5	200	1000

[7]:

```
df["Revenue"].cumsum()
```

[7]:

```
0    1000
1    2000
2    3500
3    6500
4    8100
5    9300
Name: Revenue, dtype: int64
```



[8]:
```
df["Product"].value_counts()
```

[8]:
```
Product
A    3
B    2
C    1
Name: count, dtype: int64
```

[9]:
```
df.pivot_table(values="Revenue", index="Product", aggfunc="sum")
```

[9]:

	Revenue
Product	
A	3700
B	2600
C	3000

[10]:
```
df.fillna(0)
```

[10]:

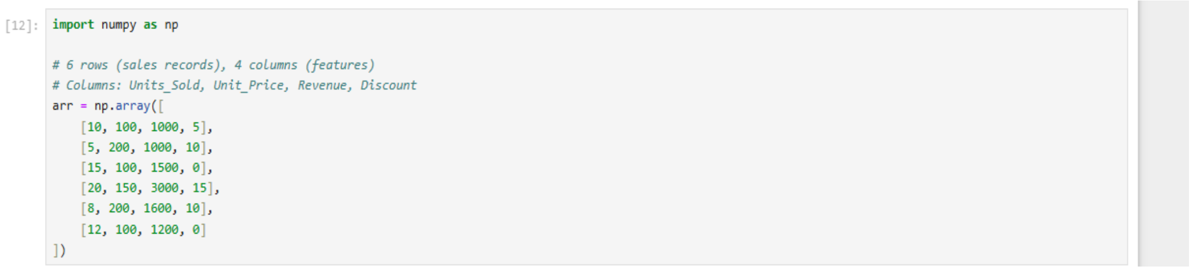
	Date	Product	Units_Sold	Unit_Price	Revenue
0	2025-01-01	A	10	100	1000
1	2025-01-02	B	5	200	1000
2	2025-01-03	A	15	100	1500
3	2025-01-04	C	20	150	3000
4	2025-01-05	B	8	200	1600
5	2025-01-06	A	12	100	1200

[11]:
```
df.rename(columns={"Units_Sold": "Units"})
```

[11]:

	Date	Product	Units	Unit_Price	Revenue
0	2025-01-01	A	10	100	1000
1	2025-01-02	B	5	200	1000
2	2025-01-03	A	15	100	1500
3	2025-01-04	C	20	150	3000
4	2025-01-05	B	8	200	1600
5	2025-01-06	A	12	100	1200

- For numpy : -
    - Dataset : -



```
[12]: import numpy as np

      # 6 rows (sales records), 4 columns (features)
      # Columns: Units_Sold, Unit_Price, Revenue, Discount
      arr = np.array([
          [10, 100, 1000, 5],
          [5, 200, 1000, 10],
          [15, 100, 1500, 0],
          [20, 150, 3000, 15],
          [8, 200, 1600, 10],
          [12, 100, 1200, 0]
      ])
```

Grains : -

1. Reshape the array to 3 rows and 8 columns

2. Find the mean of each column

3. Find rows where Units_Sold > 10

4. Add a new column for Tax = 10% of Revenue

5. Transpose the array

6. Find the maximum value in Revenue column

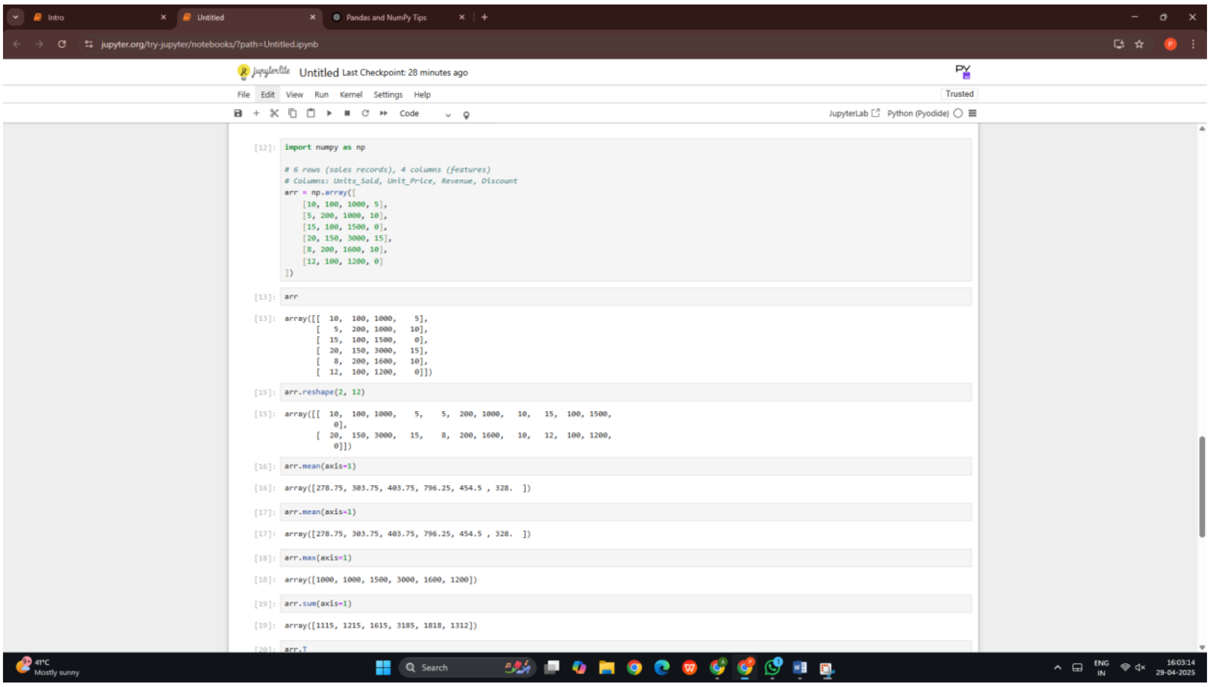
7. Normalize the Units_Sold column

8. Find rows where Discount is 0

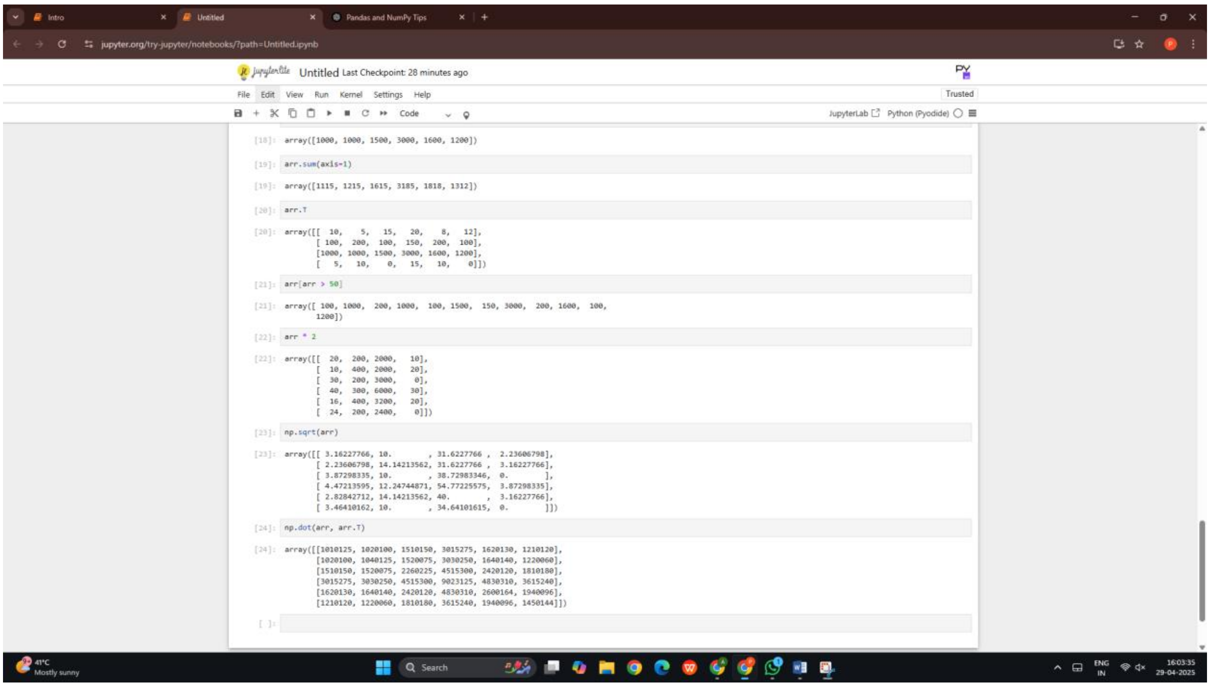
9. Find the total Revenue

10. Sort the array by Units_Sold

Output in jupyter notebook for all grains : -


Intro
Untitled
Pandas and NumPy Tips
jupyter.org/try-jupyter/notebooks/?path=Untitled.ipynb
Untitled Last Checkpoint: 28 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python (Pyodide)
41°C
Mostly sunny
Search
16:03:14
29-04-2025

```
[12]: import numpy as np

      # 6 rows (sales records), 4 columns (features)
      # Columns: Units_Sold, Unit_Price, Revenue, Discount
      arr = np.array([
          [10, 100, 1000, 5],
          [5, 200, 1000, 10],
          [15, 100, 1500, 0],
          [20, 150, 3000, 15],
          [8, 200, 1600, 10],
          [12, 100, 1200, 0]
      ])

[13]: arr

[13]: array([[  10,  100, 1000,    5],
             [   5,  200, 1000,   10],
             [  15,  100, 1500,    0],
             [  20,  150, 3000,   15],
             [   8,  200, 1600,   10],
             [  12,  100, 1200,    0]])

[15]: arr.reshape(2, 12)

[15]: array([[  10,  100, 1000,    5,    5,  200, 1000,   10,   15,  100, 1500,
                 0],
             [  20,  150, 3000,   15,    8,  200, 1600,   10,   12,  100, 1200,
                 0]])

[16]: arr.mean(axis=1)

[16]: array([278.75, 303.75, 403.75, 796.25, 454.5 , 328.  ])

[17]: arr.mean(axis=1)

[17]: array([278.75, 303.75, 403.75, 796.25, 454.5 , 328.  ])

[18]: arr.max(axis=1)

[18]: array([1000, 1000, 1500, 3000, 1600, 1200])

[19]: arr.sum(axis=1)

[19]: array([1115, 1215, 1615, 3185, 1818, 1312])

[20]: arr.T
```


Intro
Untitled
Pandas and NumPy Tips
jupyter.org/try-jupyter/notebooks/?path=Untitled.ipynb
Untitled Last Checkpoint: 28 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python (Pyodide)
41°C
Mostly sunny
Search
16:03:35
29-04-2025

```
[18]: array([1000, 1000, 1500, 3000, 1600, 1200])

[19]: arr.sum(axis=1)

[19]: array([1115, 1215, 1615, 3185, 1818, 1312])

[20]: arr.T

[20]: array([[  10,    5,   15,   20,    8,   12],
             [ 100,  200,  100,  150,  200,  100],
             [1000, 1000, 1500, 3000, 1600, 1200],
             [   5,   10,    0,   15,   10,    0]])

[21]: arr[arr > 50]

[21]: array([ 100, 1000,  200, 1000,  100, 1500,  150, 3000,  200, 1600,  100,
             1200])

[22]: arr * 2

[22]: array([[  20,  200, 2000,   10],
             [  10,  400, 2000,   20],
             [  30,  200, 3000,    0],
             [  40,  300, 6000,   30],
             [  16,  400, 3200,   20],
             [  24,  200, 2400,    0]])

[23]: np.sqrt(arr)

[23]: array([[ 3.16227766, 10.        , 31.6227766 ,  2.23606798],
             [ 2.23606798, 14.14213562, 31.6227766 ,  3.16227766],
             [ 3.87298335, 10.        , 38.72983346,  0.        ],
             [ 4.47213595, 12.24744871, 54.77225575,  3.87298335],
             [ 2.82842712, 14.14213562, 40.        ,  3.16227766],
             [ 3.46410162, 10.        , 34.64101615,  0.        ]])

[24]: np.dot(arr, arr.T)

[24]: array([[1010125, 1020100, 1510150, 3015275, 1620130, 1210120],
             [1020100, 1040125, 1520075, 3030250, 1640140, 1220060],
             [1510150, 1520075, 2260225, 4515300, 2420120, 1810180],
             [3015275, 3030250, 4515300, 9023125, 4830310, 3615240],
             [1620130, 1640140, 2420120, 4830310, 2600164, 1940096],
             [1210120, 1220060, 1810180, 3615240, 1940096, 1450144]])

[ ]:
```