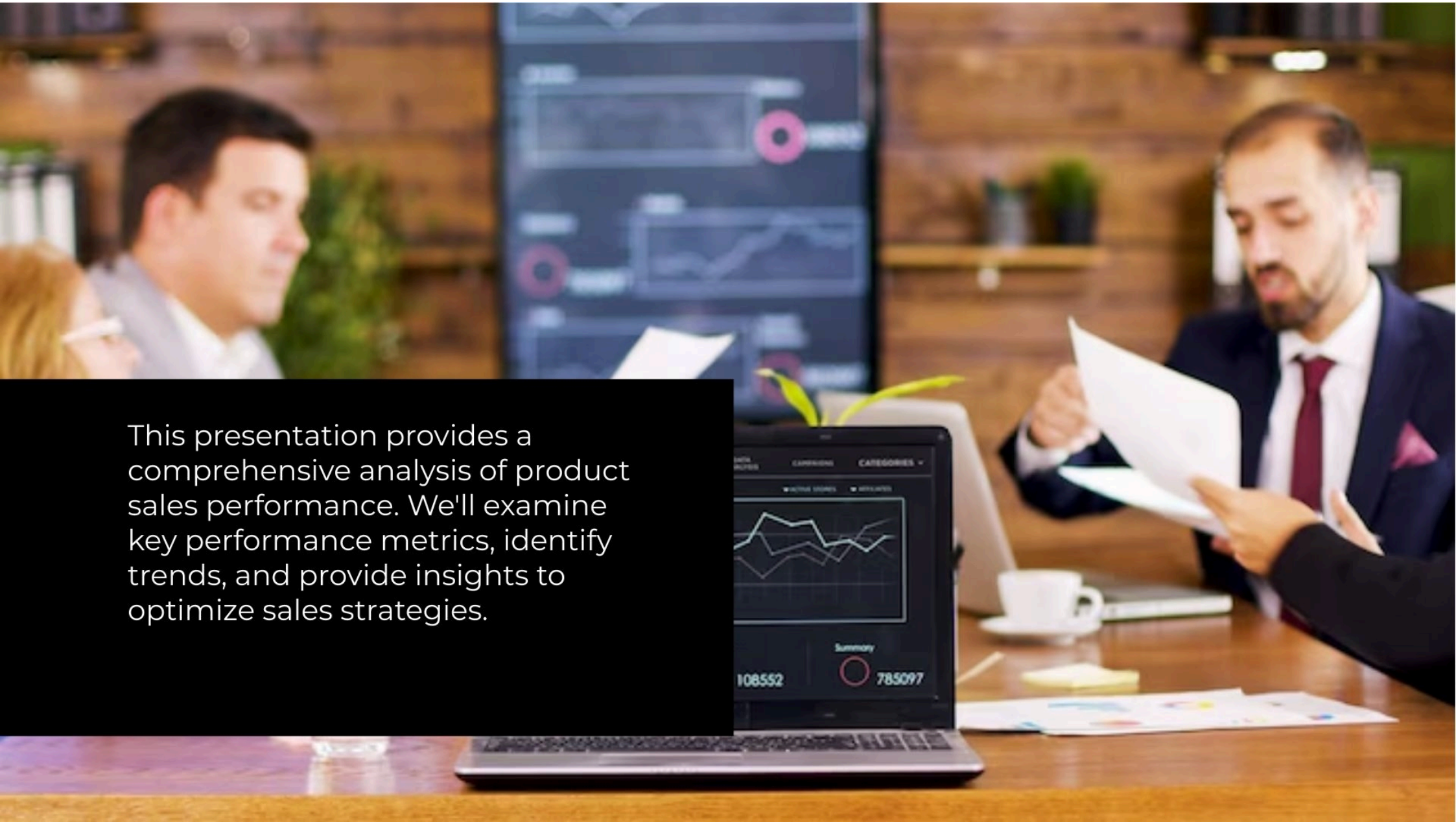
Product And Sales Analysis



This presentation provides a comprehensive analysis of product sales performance. We'll examine key performance metrics, identify trends, and provide insights to optimize sales strategies.

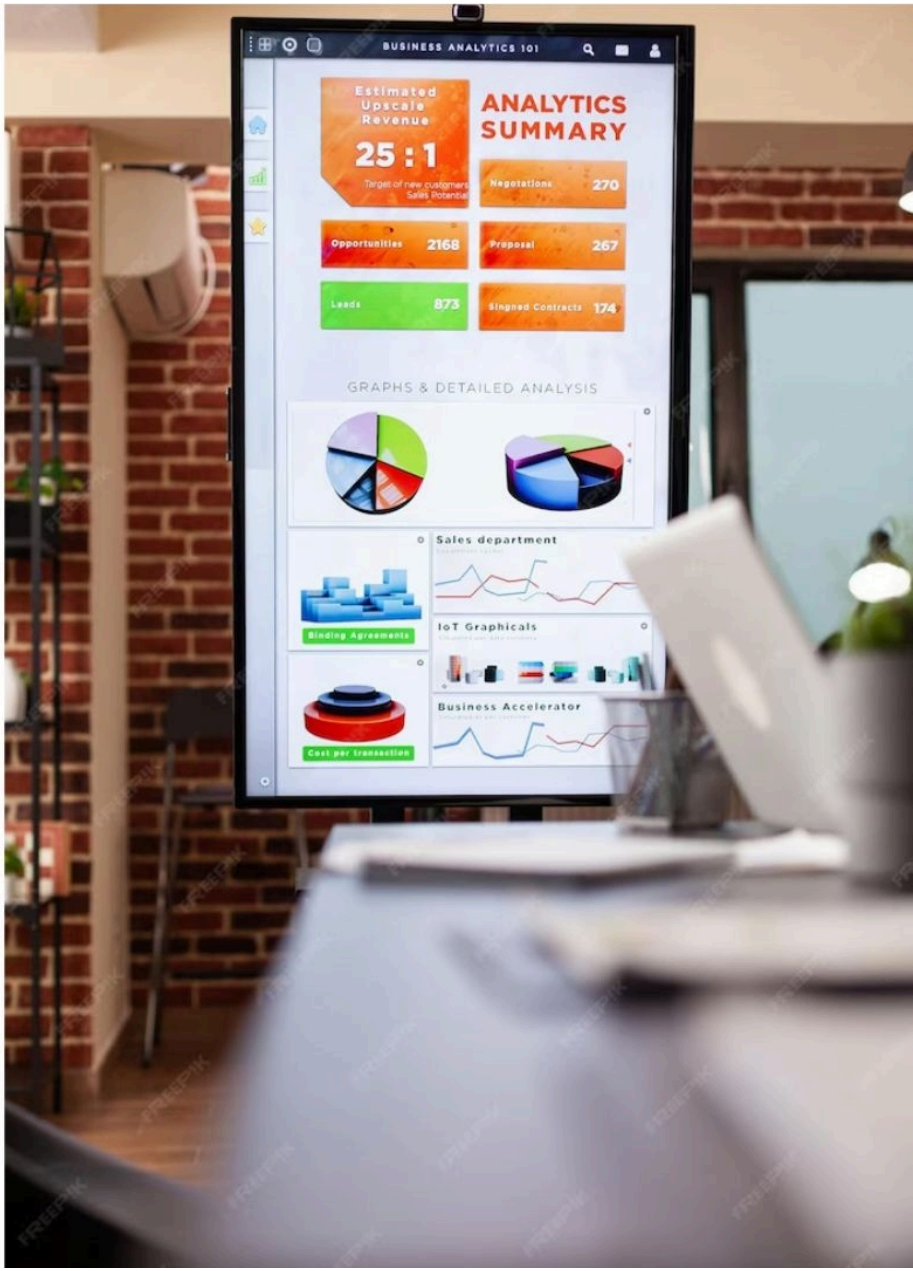
Sales Performance Metrics

01. About the project

02. Project Timeline

03. Defining a target

04. Where we are




We'll identify sales trends by analyzing historical sales data. We'll look for patterns in product sales, customer behavior, and market conditions to gain insights into what's driving sales performance.

Step 1:

First, We need to create a Dataframe of the dataset, and even before that certain libraries have to be imported.

Python3

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

Order_Details = pd.read_csv('Order.
```

Output:

	Name	Email	Product	Transaction Date
0	PERSON_1	PERSON_1@gmail.com	PRODUCT_75	01/03/2021 00:47:26
1	PERSON_2	PERSON_2@tataprojects.com	PRODUCT_75	01/03/2021 02:04:07
2	PERSON_3	PERSON_3@gmail.com	PRODUCT_63	01/03/2021 09:10:43
3	PERSON_4	PERSON_4@gmail.com	PRODUCT_63	01/03/2021 09:49:48
4	PERSON_5	PERSON_5@gmail.com	PRODUCT_34,PRODUCT_86,PRODUCT_57,PRODUCT_89	01/03/2021 10:56:46
...	...	...	...	...
576	PERSON_522	PERSON_522@gmail.com	PRODUCT_48,PRODUCT_80,PRODUCT_71,PRODUCT_68,PR...	07/03/2021 23:53:03
577	PERSON_523	PERSON_523@gmail.com	PRODUCT_8	07/03/2021 23:55:01
578	PERSON_523	PERSON_523@gmail.com	PRODUCT_36,PRODUCT_14,PRODUCT_64,PRODUCT_28,PR...	07/03/2021 23:58:24
579	PERSON_524	PERSON_524@gmail.com	PRODUCT_75,PRODUCT_71,PRODUCT_86,PRODUCT_63,PR...	07/03/2021 23:59:26
580	PERSON_525	PERSON_525@gmail.com	PRODUCT_66,PRODUCT_34	07/03/2021 23:59:19

581 rows × 4 columns

Step 2:

Create a new column called Time that has the DateTime format after converting the Transaction Date column into it. The DateTime format, which has the pattern **YYYY-MM-DD HH:MM:SS**, can be customized however you choose. Here we're more interested in obtaining hours, so we can have an Hour column by using an in-built function for the same:

Python3

```
# here we have taken Transaction
# date column
Order_Details['Time'] = pd.to_date

# After that we extracted hour
# from Transaction date column
Order_Details['Hour'] = (Order_Det
```

Step 3:

We then require the “n” busiest hours. For that, we get the first “n” entries in a list containing the occurrence rates of the hours when the transaction took place. To further simplify the manipulation of the provided data in Python, we may utilize value counts for frequencies and tolist() to convert to list format. We are also compiling a list of the associated index values.

Python3

```
# n =24 in this case, can be modif
# as per need to see top 'n' busie
timemost1 = Order_Details['Hour'].

timemost2 = Order_Details['Hour'].
```

Step 4:

Finally, we stack the indices (hour) and frequencies together to yield the final result.

Python3

```
tmost = np.column_stack((timemost1

print(" Hour Of Day" + "\t" + "Cum
print('\n'.join('\t\t'.join(map(st
```

Hour Of Day	Cumulative Number of Purchases
23	51
12	51
22	45
19	42
21	41
15	41
20	39
11	37
13	33
18	33
16	29
14	28
17	27
10	24
0	17
9	14
8	10
7	6
1	4
2	3
5	3
6	2
3	1

Step 5:

Before we can create an appropriate data visualization, we must make the list slightly more customizable. To do so, we gather the hourly frequencies and perform the following tasks:

Python3

```
timemost = Order_Details['Hour'].v
timemost1 = []

for i in range(0,23):
    timemost1.append(i)

timemost2 = timemost.sort_index()
timemost2.tolist()
timemost2 = pd.DataFrame(timemost2
```

Step 6:

For data visualization, we will proceed with Matplotlib for better comprehensibility, as it is one of the most convenient and commonly used libraries. But, It is up to you to choose any of the pre-existing libraries like Matplotlib, Ggplot, Seaborn, etc., to plot the data graphically.

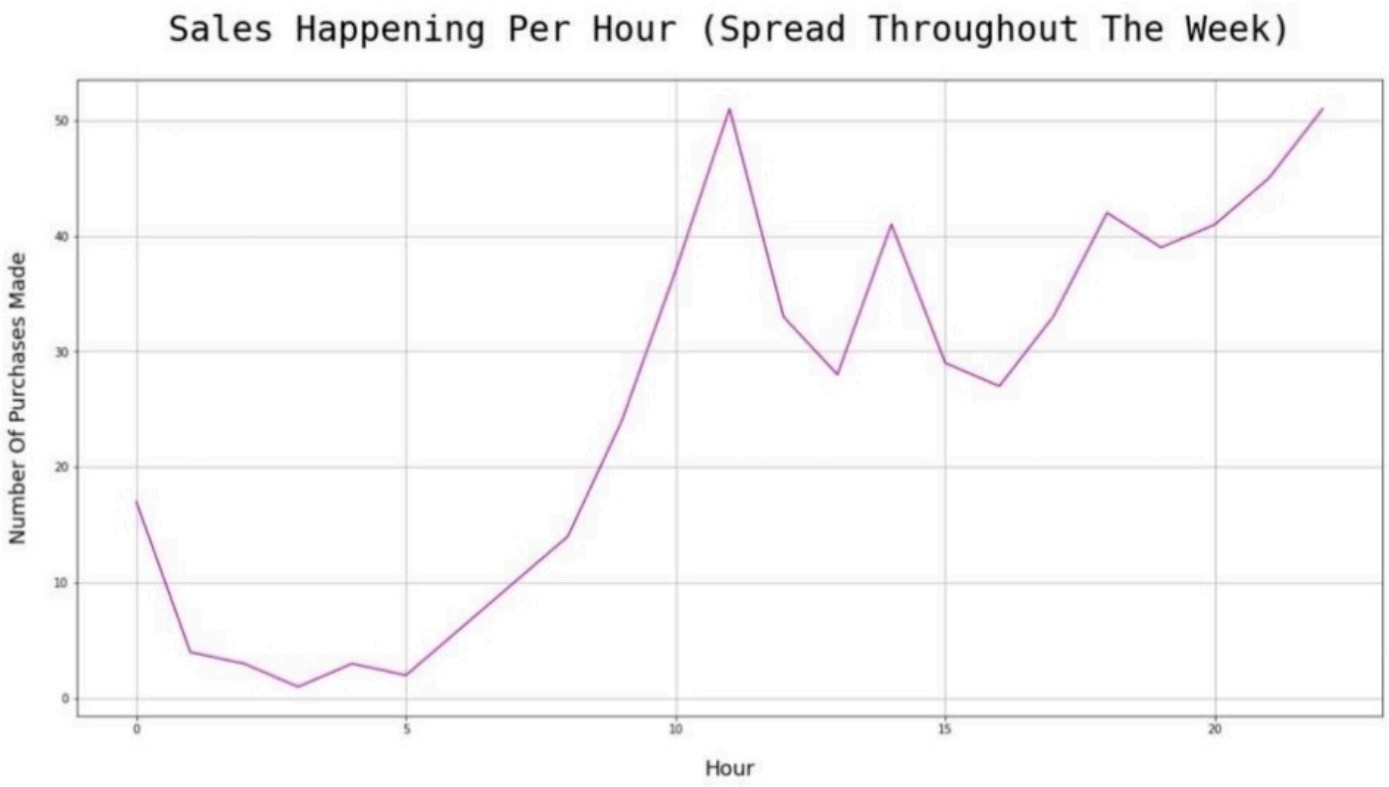
The commands written below are mainly to ensure that X-axis takes up the values of hours and Y-axis takes up the importance of the number of transactions affected, and also various other aspects of a line chart, including color, font, etc., to name a few.

Python3

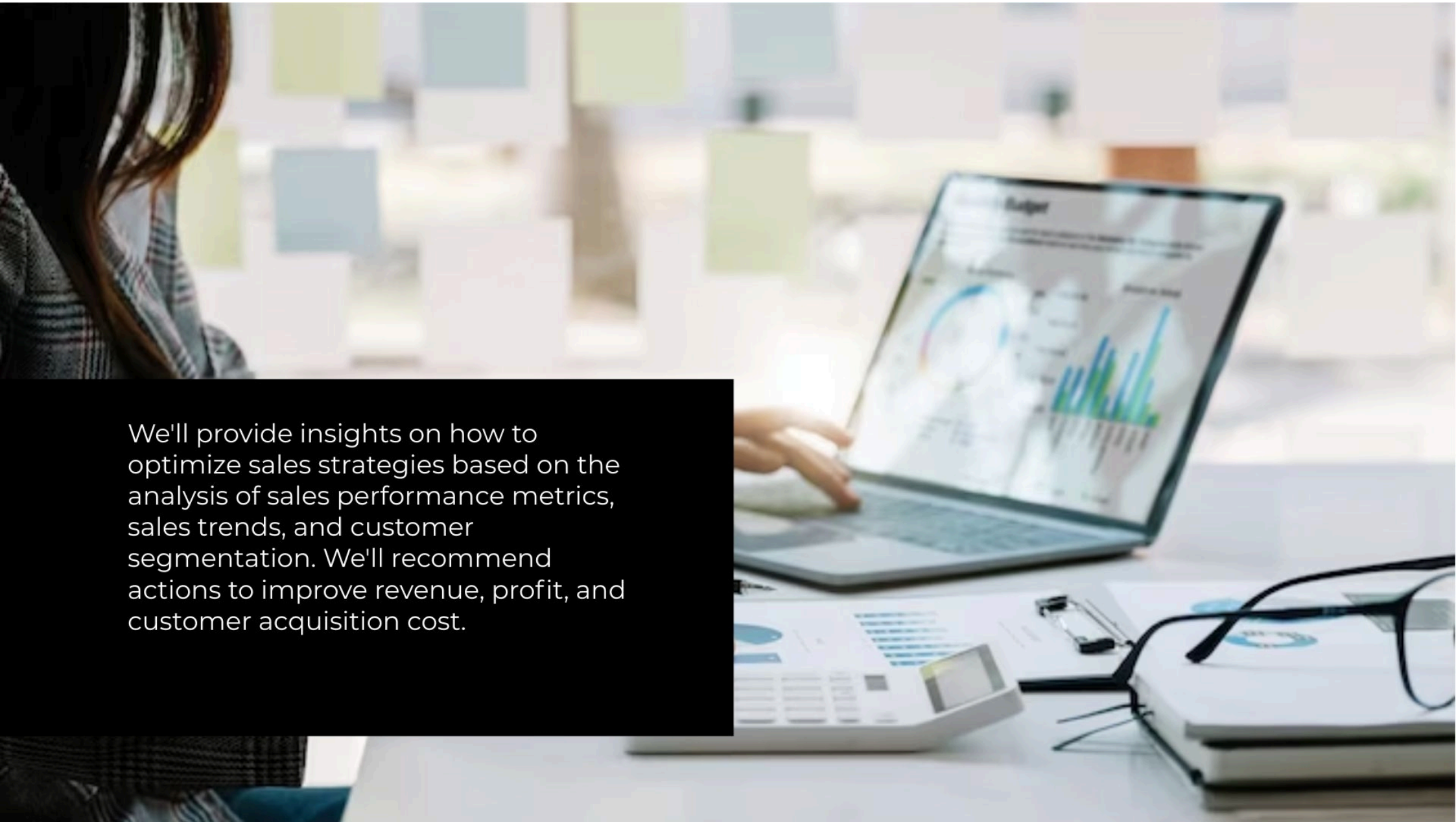
```
plt.figure(figsize=(20, 10))

plt.title('Sales Happening Per Hou
          fontdict={'fontname': 'm

plt.ylabel("Number Of Purchases Ma
plt.xlabel("Hour", fontsize=18, la
plt.plot(timemost1, timemost2, col
plt.grid()
plt.show()
```



The results are indicative of how sales typically peak in late evening hours prominently, and this data can be incorporated into business decisions to promote a product during that time specifically.



We'll provide insights on how to optimize sales strategies based on the analysis of sales performance metrics, sales trends, and customer segmentation. We'll recommend actions to improve revenue, profit, and customer acquisition cost.


THANK YOU