AI ASSISTED CODING

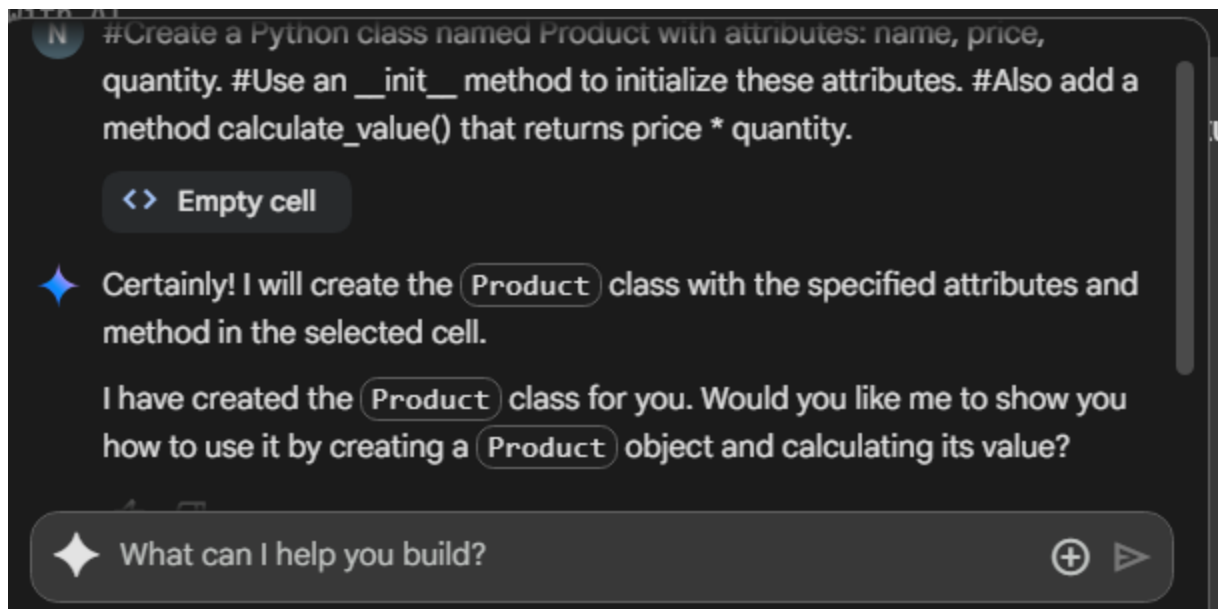## LAB ASSIGNMENT – 6.5

SATHVIKA CH
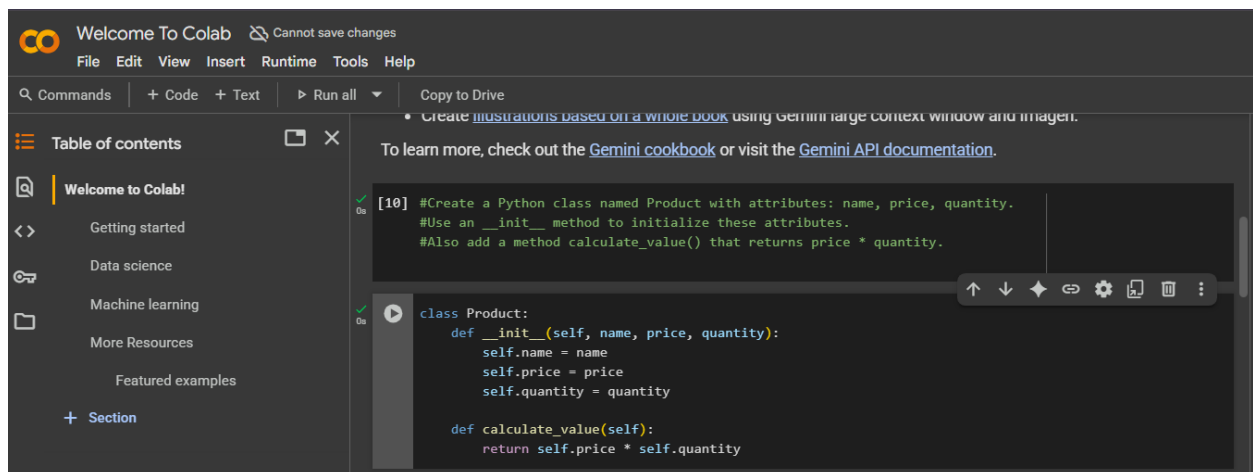
2403A53026

24BTCAICYB01-2-1

#TASK-1

Class Design Using AI Assistance:
• Begin defining a Product class with attributes: name, price, quantity.
• Use the AI suggestion feature to automatically complete the __init__() method.
• Add a method calculate_value() to return price * quantity.

PROMPT:

#Create a Python class named Product with attributes: name, price, quantity. #Use an __init__ method to initialize these attributes. #Also add a method calculate_value() that returns price * quantity.

Certainly! I will create the `Product` class with the specified attributes and method in the selected cell.

I have created the `Product` class for you. Would you like me to show you how to use it by creating a `Product` object and calculating its value?

What can I help you build?

OUTPUT:

• Create illustrations based on a whole book using Gemini large context window and Imagen.

To learn more, check out the Gemini cookbook or visit the Gemini API documentation.

```python
[10] #Create a Python class named Product with attributes: name, price, quantity.
     #Use an __init__ method to initialize these attributes.
     #Also add a method calculate_value() that returns price * quantity.
```

```python
class Product:
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity

    def calculate_value(self):
        return self.price * self.quantity
```
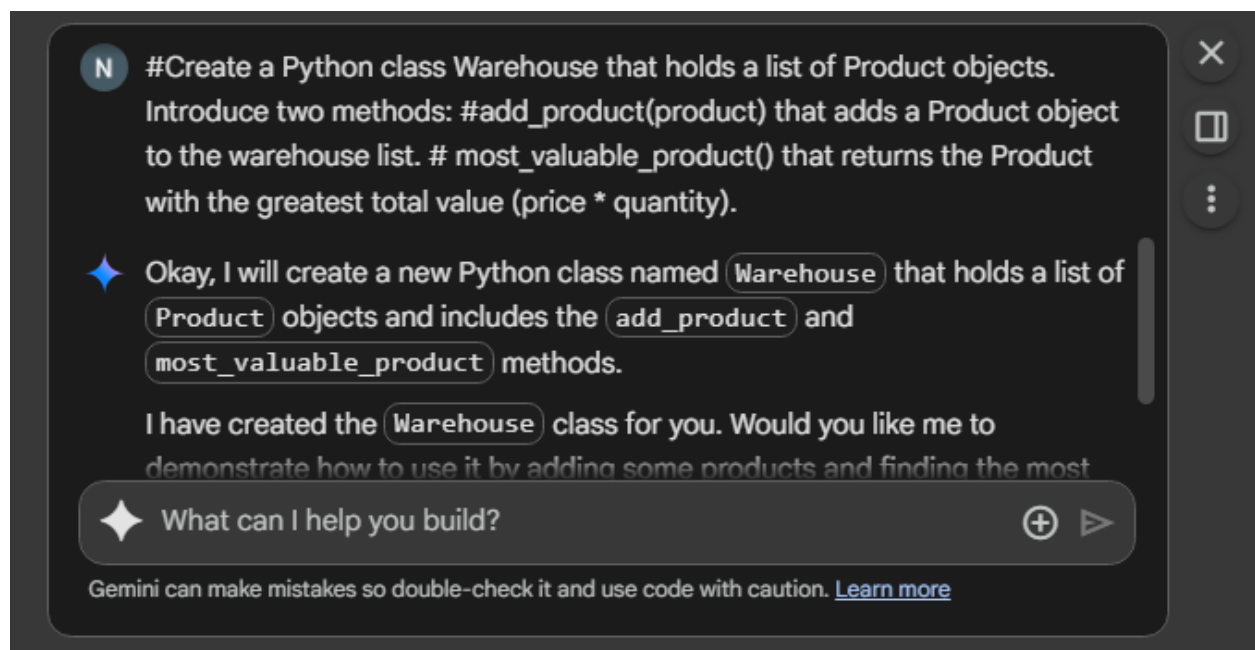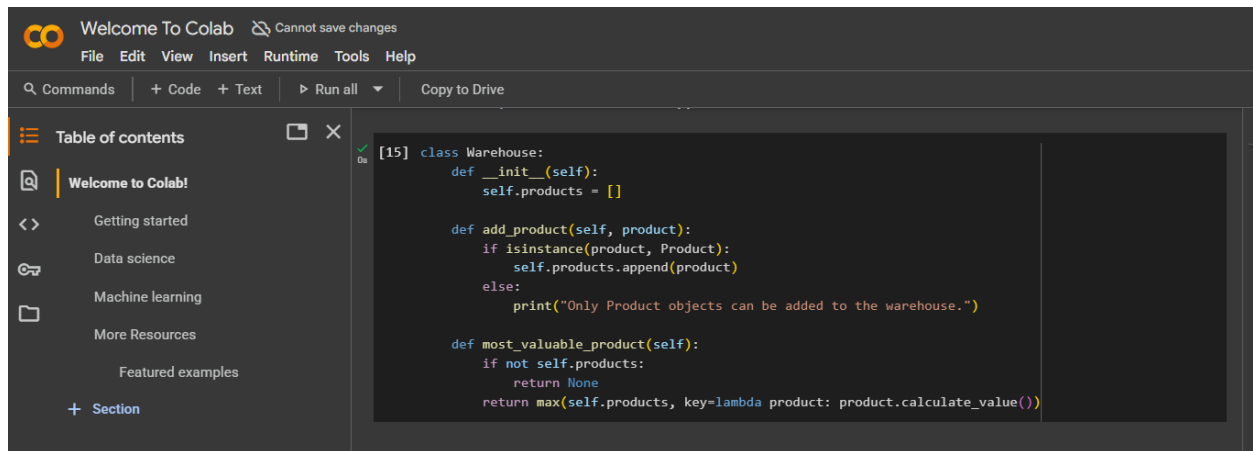
#TASK-2

Define a Warehouse class with a list of Product objects.
• Use code completion to help implement:
o A method to add a product.
o A method to display the most valuable product.

PROMPT:

N #Create a Python class Warehouse that holds a list of Product objects.
Introduce two methods: #add_product(product) that adds a Product object
to the warehouse list. # most_valuable_product() that returns the Product
with the greatest total value (price * quantity).

Okay, I will create a new Python class named `Warehouse` that holds a list of
`Product` objects and includes the `add_product` and
`most_valuable_product` methods.

I have created the `Warehouse` class for you. Would you like me to
demonstrate how to use it by adding some products and finding the most

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. Learn more

OUTPUT:

```
[15] class Warehouse:
         def __init__(self):
             self.products = []

         def add_product(self, product):
             if isinstance(product, Product):
                 self.products.append(product)
             else:
                 print("Only Product objects can be added to the warehouse.")

         def most_valuable_product(self):
             if not self.products:
                 return None
             return max(self.products, key=lambda product: product.calculate_value())
```

SUMMARY:

In this task, I had to create two classes in Python: Product and Warehouse. The Product class had name, price, and quantity, and a method to calculate total value. The Warehouse class had to keep a list of products, allow adding new products, and show the most valuable product.

I used Git Hub Copilot for this. When I started writing the code, Copilot gave me suggestions automatically. It was a useful and time-saving experience.