Classes and Object

```
class Customer:
    def _init_(self, customer_id, first_name, last_name, phone_no,
email):
        self.customer_id = customer_id
        self.first_name = first_name
        self.last_name = last_name
        self.phone_no = phone_no
        self.email = email

class CustAddress:
    def _init_(self, customer_id, state, city, zipcode, lane_no,
house_no):
        self.customer_id = customer_id
        self.state = state
        self.city = city
        self.zipcode = zipcode
        self.lane_no = lane_no
        self.house_no = house_no

class Location:
    def _init_(self, location_id, location_name, address, type,
capacity):
        self.location_id = location_id
        self.location_name = location_name
        self.address = address
        self.type = type
        self.capacity = capacity

class Supplier:
    def _init_(self, supplier_id, name, contact):
        self.supplier_id = supplier_id
        self.name = name
        self.contact = contact

class Product:
    def _init_(self, product_id, category, descr, brand, price,
units):
        self.product_id = product_id
        self.category = category
        self.descr = descr
        self.brand = brand
        self.price = price
        self.units = units

class Order:
    def _init_(self, order_id, customer_id, location_id, order_date,
```

```
shipping_date, delivered_date):
        self.order_id = order_id
        self.customer_id = customer_id
        self.location_id = location_id
        self.order_date = order_date
        self.shipping_date = shipping_date
        self.delivered_date = delivered_date

class Payment:
    def _init_(self, order_id, product_name, product_category,
address, product_id, discount, total_price, quantity, payment_mode,
coupon_applied, delivery_status):
        self.order_id = order_id
        self.product_name = product_name
        self.product_category = product_category
        self.address = address
        self.product_id = product_id
        self.discount = discount
        self.total_price = total_price
        self.quantity = quantity
        self.payment_mode = payment_mode
        self.coupon_applied = coupon_applied
        self.delivery_status = delivery_status

class Review:
    def _init_(self, customer_id, product_id, review, rating):
        self.customer_id = customer_id
        self.product_id = product_id
        self.review = review
        self.rating = rating

#customers:
customer1 = Customer("C001", "Balaji", "Solunke", 1234567890,
"balajisolunke62@gmail.com")
customer2 = Customer("C002", "Manoj", "Kudkyal", 9087654321,
"manojkudkyal123@gmail.com")

#CustAddress:
address1 = CustAddress("C001", "Maharashtra", "Solapur", "413522",
"C.nagar,22", "42")
address2 = CustAddress("C002", "Maharashtra", "Latur", "411006",
"7717,Laxmi nagar", "2A")

#Location:
store1 = Location("L001", "Downtown Store", "123 Main St, Latur, CA
94105", "Store", 500)
warehouse1 = Location("L002", "Central Warehouse", "10 Industrial Ave,
Solapur, CA 94608", "Warehouse", 1000)

#Supplier:
```

```
supplier1 = Supplier("S001", "Acme Electronics", "1-800-555-1212")
supplier2 = Supplier("S002", "Best Products Inc.", "1-800-555-2323")

#Product:
product1 = Product("P001", "Electronics", "Smartphone", "Apple",
799.99, 50)
product2 = Product("P002", "Clothing", "T-shirt", "Nike", 25.00, 100)

#Order:
order1 = Order("O001", "C001", "L001", "2024-01-02", "2024-01-05",
"2024-01-07")
order2 = Order("O002", "C002", "L001", "2024-01-03", "2024-01-06",
"2024-01-08")

#Payment:
payment1 = Payment("O001", "Smartphone", "Electronics", "P001", 10.0,
789.99, 1, "Credit card", "NOCOUPON", "Delivered")
payment2 = Payment("O002", "T-shirt", "Clothing", "P002", 0.0, 50.00,
2, "PayPal", "WELCOME10", "Shipped")

#Review:
review1 = Review("C001", "P001", "Great phone, fast and reliable!", 5)
review2 = Review("C002", "P002", "Comfortable and stylish T-shirt.",
4)

import pandas as pd

df2 = pd.read_csv("C:\\Users\\Sakshi\\OneDrive\\Documents\\
DataCoSupplyChainDataset.csv")

import pandas as pd

class Product:
    def __init__(self, product_data):
        self.df = product_data

    def display_product_info(self, product_index):
        product_info = self.df.iloc[product_index]
        print("Product Information:")
        print(f"Product Name: {product_info['Product Name']}")
        print(f"Product Description: {product_info['Product
Description']}")
        print(f"Product Price: {product_info['Product Price']}")
        print(f"Shipping Date: {product_info['shipping date
(DateOrders)']}")
        print(f"Shipping Mode: {product_info['Shipping Mode']}")
        print("\n")

    def check_product_availability(self, product_name):
        available_products = self.df.loc[self.df['Product Name'] ==
product_name]
```

```
        return not available_products.empty

    def calculate_average_price(self, category_id):

        category_products = self.df.loc[self.df['Product Category Id'] == category_id]
        average_price = category_products['Product Price'].mean()
        return average_price


# you have a DataFrame df2 containing the specified columns
product_instance = Product(df2)

# Example usage of the methods
# Display information for the product of given index
product_instance.display_product_info(100)


# Check availability of a product
availability = product_instance.check_product_availability('Field & Stream Sportsman 16 Gun Fire Safe')  # Check availability of a product
print(f"Is Field & Stream Sportsman 16 Gun Fire Safe available? {availability}")

# Calculate average price for products with Category Id 73
avg_price = product_instance.calculate_average_price(73)  # Calculate average price for products with Category Id 123
print(f"Average price for Category Id 73: ${avg_price}")
```

```
Product Information:
Product Name: Nike Men's Dri-FIT Victory Golf Polo
Product Description: nan
Product Price: 50.0
Shipping Date: 02-09-2017 01:01
Shipping Mode: Standard Class


Is Field & Stream Sportsman 16 Gun Fire Safe available? True
Average price for Category Id 73: $327.75
```

```
import pandas as pd

class Customer:
    def __init__(self, customer_data):

        self.df = customer_data

    def display_customer_info(self, customer_index):
```

```
        customer_info = self.df.iloc[customer_index]
        print("Customer Information:")
        print(f"Customer ID: {customer_info['Customer Id']}")
        print(f"Name: {customer_info['Customer Fname']}
{customer_info['Customer Lname']}")
        print(f"Email: {customer_info['Customer Email']}")
        print(f"Address: {customer_info['Customer Street']},
{customer_info['Customer City']}, {customer_info['Customer State']}
{customer_info['Customer Zipcode']}")
        print("\n")

    def is_customer_from_country(self, country):

        customers_from_country = self.df.loc[self.df['Customer
Country'] == country]
        return not customers_from_country.empty

    def get_customer_full_name(self, customer_id):

        customer_name = self.df.loc[self.df['Customer Id'] ==
customer_id, ['Customer Fname', 'Customer Lname']]
        if not customer_name.empty:
            return f"{customer_name['Customer Fname'].values[0]}
{customer_name['Customer Lname'].values[0]}"
        else:
            return "Customer not found."

# you have a DataFrame df2 containing the specified columns
customer_instance = Customer(df2)

# Example usage of the methods
customer_instance.display_customer_info(0)  # Display information for
the first customer
from_country = customer_instance.is_customer_from_country('United
States')  # Check if a customer is from a specific country
print(f"Is the customer from United States? {from_country}")

customer_name = customer_instance.get_customer_full_name(123)  # Get
the full name of a customer by ID
print(f"Full name of Customer ID 123: {customer_name}")
```

```
Customer Information:
Customer ID: 20755
Name: Cally Holloway
Email: XXXXXXXXX
Address: 5365 Noble Nectar Island, Caguas, PR 725.0


Is the customer from United States? False
Full name of Customer ID 123: Mary Mann
```

```
import pandas as pd

class Order:
    def __init__(self, order_data):

        self.df = order_data

    def display_order_info(self, order_index):

        order_info = self.df.iloc[order_index]
        print("Order Information:")
        print(f"Order ID: {order_info['Order Id']}")
        print(f"Customer ID: {order_info['Order Customer Id']}")
        print(f"Order Date: {order_info['order date (DateOrders)']}")
        print(f"City: {order_info['Order City']}")
        print(f"State: {order_info['Order State']}")
        print(f"Country: {order_info['Order Country']}")
        print(f"Region: {order_info['Order Region']}")
        print(f"Order Status: {order_info['Order Status']}")
        print("\n")

    def calculate_total_sales(self, order_id):

        order_sales = self.df.loc[self.df['Order Id'] == order_id,
'Sales'].sum()
        return order_sales

    def check_order_status(self, order_id):

        order_status = self.df.loc[self.df['Order Id'] == order_id,
'Order Status'].values
        return order_status[0] if len(order_status) > 0 else "Order
not found."

# you have a DataFrame df2 containing the specified columns
order_instance = Order(df2)

# Example usage of the methods
order_instance.display_order_info(0)  # Display information for the
first order
total_sales = order_instance.calculate_total_sales(456)  # Calculate
total sales for a specific order
print(f"Total sales for Order ID 456: ${total_sales:.2f}")

order_status = order_instance.check_order_status(789)  # Check the
status of a specific order
print(f"Order ID 789 status: {order_status}")

Order Information:
Order ID: 77202
```

```
Customer ID: 20755
Order Date: 1/31/2018 22:56
City: Bekasi
State: Java Occidental
Country: Indonesia
Region: Southeast Asia
Order Status: COMPLETE


Total sales for Order ID 456: $699.86
Order ID 789 status: COMPLETE
```