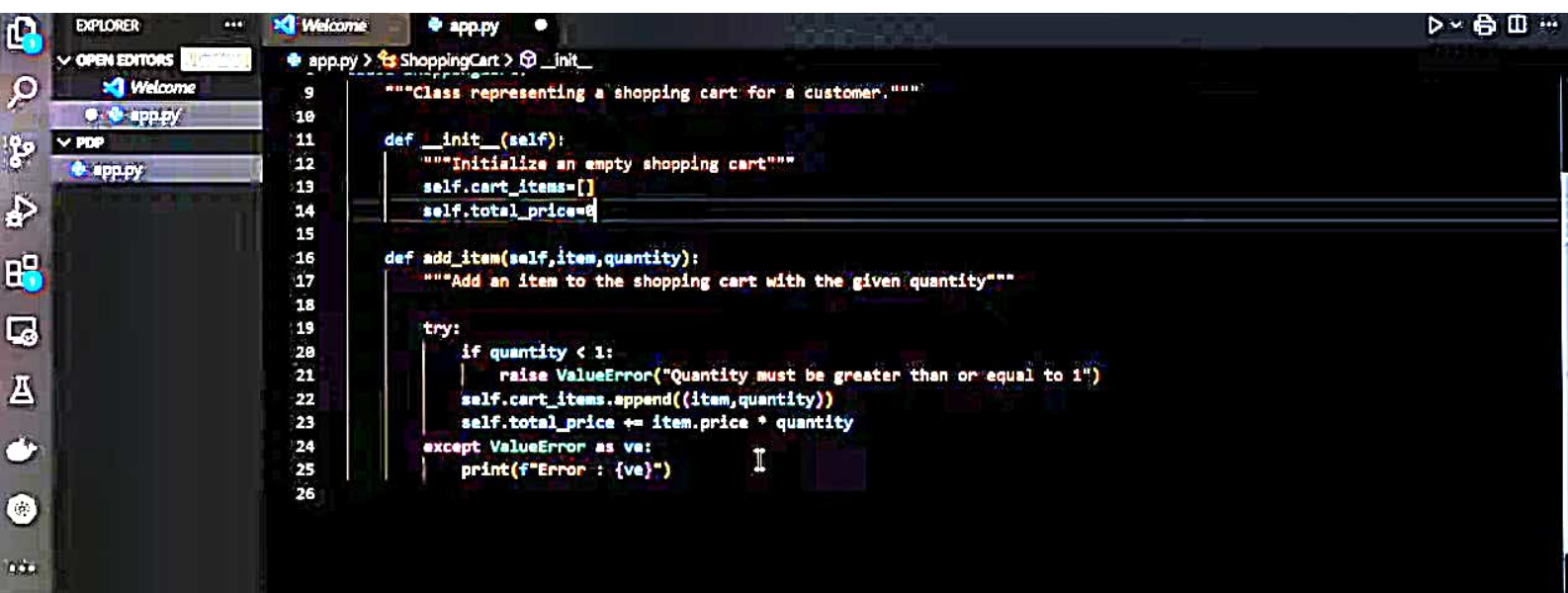


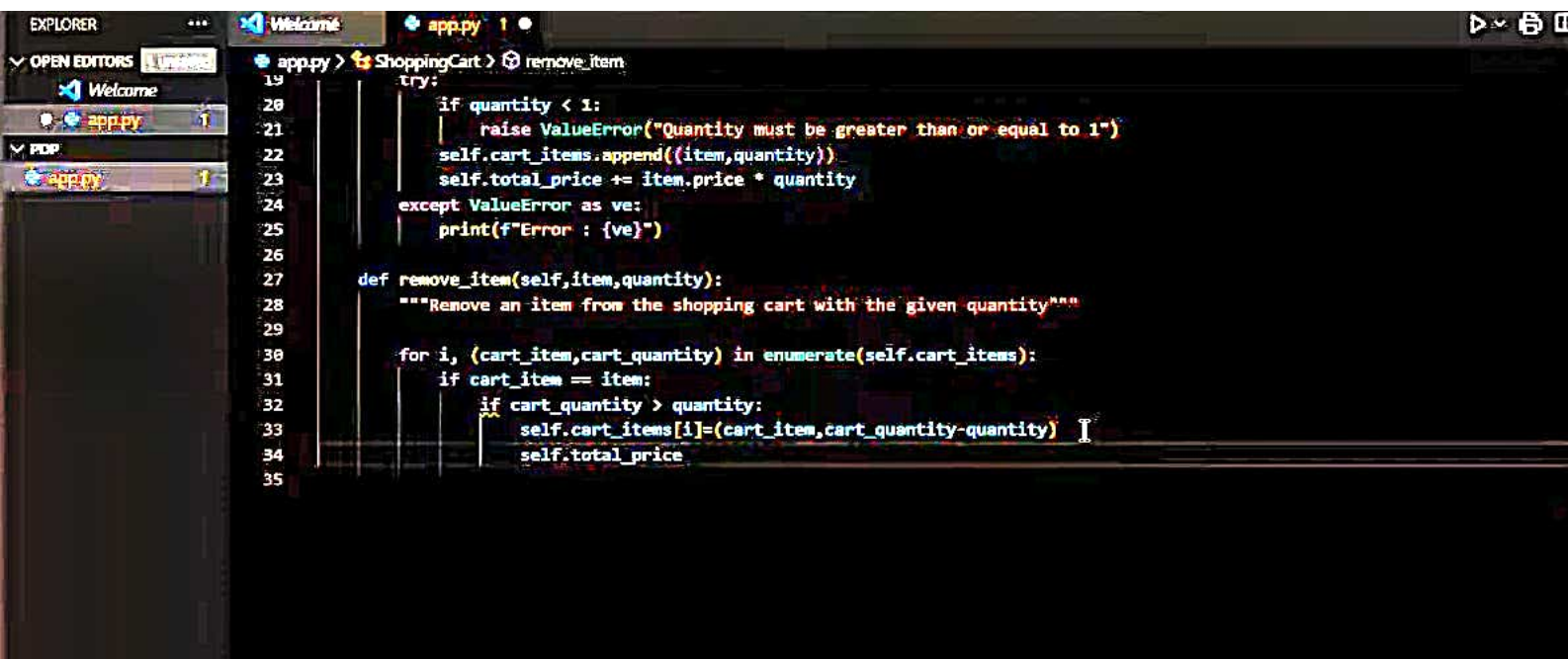
The image shows a code editor with a dark theme. On the left, there is a sidebar with a file explorer and a list of open editors. The main area displays a Python file named `app.py` with a class definition for `ShoppingCart`. The code is as follows:

```
1 2
3  #class-ShoppingCart,Item,Customer
4  #OnlineStore
5
6  #ShoppingCart
7
8  class ShoppingCart:
9      """Class representing a shopping cart for a customer."""
10     def __init__(self):
11         """Initialize an empty shopping cart"""
12         self.cart_items=[]
13         self.total_price=0
14
15     def add_item(self,item,quantity):
16         """Add an item to the shopping cart with the given quantity"""
17
18     try:
19         if quantity<1:
20
```



The image shows a Visual Studio Code editor window with a dark theme. On the left, the Explorer sidebar shows a file named 'app.py' under a folder named 'PDP'. The main editor area displays the code for 'app.py', which defines a 'ShoppingCart' class. The code includes a docstring, an '__init__' method to initialize an empty cart and a total price of 0, and an 'add_item' method that appends items to the cart and updates the total price, with a try-except block for handling 'ValueError' exceptions when the quantity is less than 1.

```
9      """Class representing a shopping cart for a customer."""
10
11      def __init__(self):
12          """Initialize an empty shopping cart"""
13          self.cart_items=[]
14          self.total_price=0
15
16      def add_item(self,item,quantity):
17          """Add an item to the shopping cart with the given quantity"""
18
19          try:
20              if quantity < 1:
21                  raise ValueError("Quantity must be greater than or equal to 1")
22              self.cart_items.append((item,quantity))
23              self.total_price += item.price * quantity
24          except ValueError as ve:
25              print(f"Error : {ve}")
26
```



```
19 try:
20     if quantity < 1:
21         raise ValueError("Quantity must be greater than or equal to 1")
22     self.cart_items.append((item, quantity))
23     self.total_price += item.price * quantity
24 except ValueError as ve:
25     print(f"Error : {ve}")
26
27 def remove_item(self, item, quantity):
28     """Remove an item from the shopping cart with the given quantity"""
29
30     for i, (cart_item, cart_quantity) in enumerate(self.cart_items):
31         if cart_item == item:
32             if cart_quantity > quantity:
33                 self.cart_items[i] = (cart_item, cart_quantity - quantity)
34                 self.total_price
35
```



```
app.py > ShoppingCart > remove_item
22     self.cart_items.append((item, quantity))
23     self.total_price += item.price * quantity
24 except ValueError as ve:
25     print(f"Error : {ve}")
26
27 def remove_item(self, item, quantity):
28     """Remove an item from the shopping cart with the given quantity"""
29
30     for i, (cart_item, cart_quantity) in enumerate(self.cart_items):
31         if cart_item == item:
32             if cart_quantity > quantity:
33                 self.cart_items[i] = (cart_item, cart_quantity - quantity)
34                 self.total_price -= item.price * quantity
35
36             elif cart_quantity == quantity:
37                 self.cart_items.pop(i)
38                 self.total_price -= item.price * quantity
39         else:
40             print(f"Error: Cannot remove {quantity} {item.name}(s), only {cart_quantity} available in cart")
41             break
42
```

```
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

def add_item(self,item,quantity):
    """Add an item to the shopping cart with the given quantity"""
    try:
        if quantity < 1:
            raise ValueError("Quantity must be greater than or equal to 1")
        self.cart_items.append((item,quantity))
        self.total_price += item.price * quantity
    except ValueError as ve:
        print(f"Error : {ve}")

def remove_item(self,item,quantity):
    """Remove an item from the shopping cart with the given quantity"""
    for i, (cart_item, cart_quantity) in enumerate(self.cart_items):
        if cart_item == item:
            if cart_quantity > quantity:
                self.cart_items[i]=(cart_item, cart_quantity-quantity)
                self.total_price -= item.price * quantity

            elif cart_quantity == quantity:
                self.cart_items.pop(i)
                self.total_price -= item.price * quantity
            else:
```


OPEN EDITORS

Welcome

app.py

PDP

app.py

> OUTLINE

```
app.py > ShoppingCart > remove_item
17. Add all items to the shopping cart with the given quantity
18
19
20     try:
21         if quantity < 1:
22             raise ValueError("Quantity must be greater than or equal to 1")
23         self.cart_items.append((item,quantity))
24         self.total_price += item.price * quantity
25     except ValueError as ve:
26         print(f"Error : {ve}")
27
28 def remove_item(self,item,quantity):
29     """Remove an item from the shopping cart with the given quantity"""
30
31     for i, (cart_item, cart_quantity) in enumerate(self.cart_items):
32         if cart_item == item:
33             if cart_quantity > quantity:
34                 self.cart_items[i]=(cart_item, cart_quantity-quantity)
35                 self.total_price -= item.price * quantity
36
37             elif cart_quantity == quantity:
38                 self.cart_items.pop(i)
39                 self.total_price -= item.price * quantity
40         else:
41             print(f"Error: Cannot remove {quantity} {item.name}(s), only {cart_quantity} available in cart")
42             break
```

OPEN EDITORS

- Welcome
- app.py
- POP
- app.py

OUTLINE

```
app.py > ShoppingCart > remove_item
17 """Add an item to the shopping cart with the given quantity"""
18
19 try:
20     if quantity < 1:
21         raise ValueError("Quantity must be greater than or equal to 1")
22     self.cart_items.append((item, quantity))
23     self.total_price += item.price * quantity
24 except ValueError as ve:
25     print(f"Error : {ve}")
26
27 def remove_item(self, item, quantity):
28     """Remove an item from the shopping cart with the given quantity"""
29
30     for i, (cart_item, cart_quantity) in enumerate(self.cart_items):
31         if cart_item == item:
32             if cart_quantity > quantity:
33                 self.cart_items[i] = (cart_item, cart_quantity - quantity)
34                 self.total_price -= (parameter) quantity: Any
35             elif cart_quantity == quantity:
36                 self.cart_items.pop(i)
37                 self.total_price -= item.price * quantity
38             else:
39                 print(f"Error: Cannot remove {quantity} {item.name}(s), only {cart_quantity} available in cart")
40                 break
41
42
```



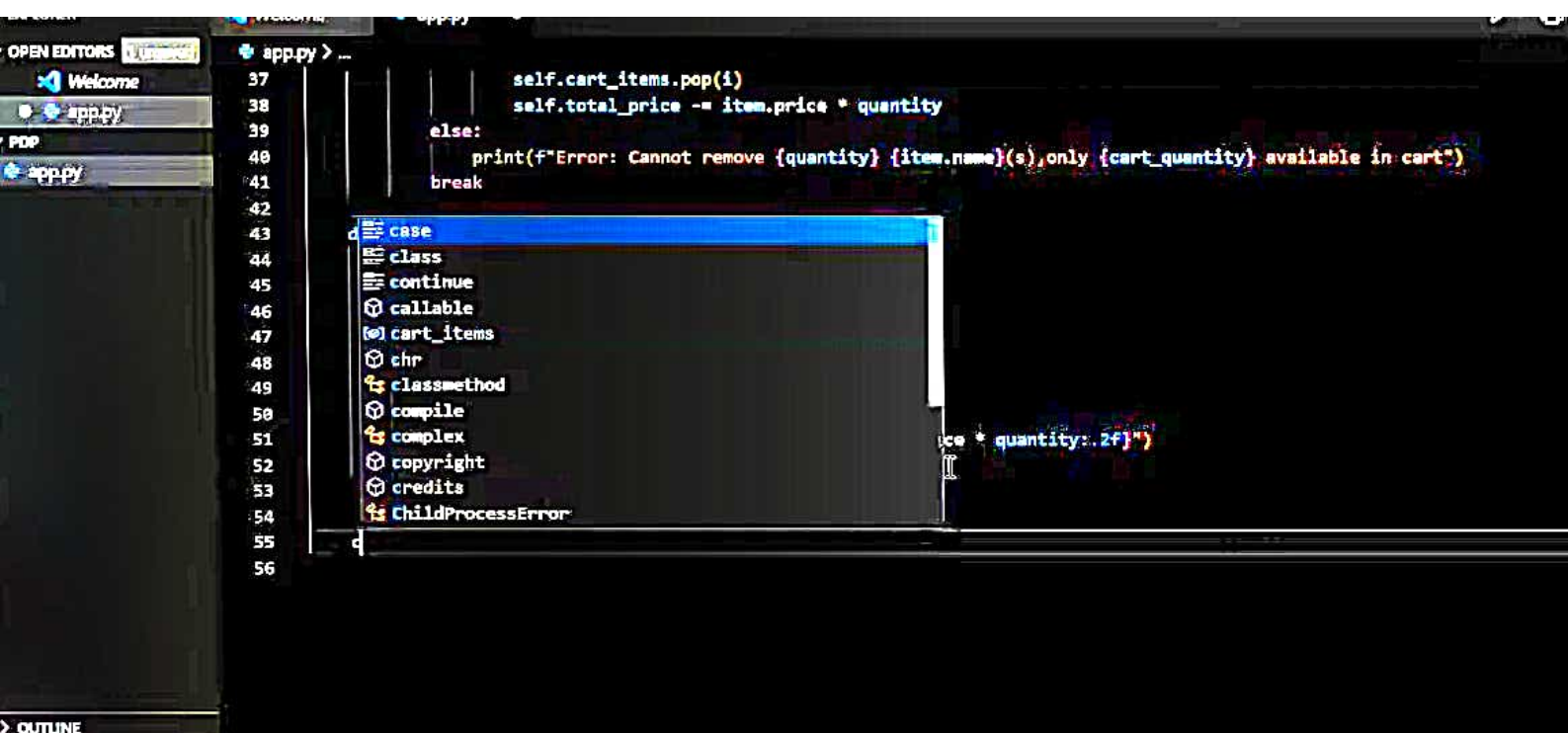

```
app.py > ShoppingCart > view_cart
Remove all items from the shopping cart with the given quantity
29
30     for i, (cart_item, cart_quantity) in enumerate(self.cart_items):
31         if cart_item == item:
32             if cart_quantity > quantity:
33                 self.cart_items[i] = (cart_item, cart_quantity - quantity)
34                 self.total_price -= item.price * quantity
35
36             elif cart_quantity == quantity:
37                 self.cart_items.pop(i)
38                 self.total_price -= item.price * quantity
39             else:
40                 print(f"Error: Cannot remove {quantity} {item.name}(s), only {cart_quantity} available in cart")
41                 break
42
43     def view_cart(self):
44         """View the current items in the shopping cart"""
45
46         if not self.cart_items:
47
```

EXPLORER
OPEN EDITORS
Welcome
app.py
app.py
app.py

```
app.py > ShoppingCart > add_item
5
6 #ShoppingCart
7
8 class ShoppingCart:
9     """Class representing a shopping cart for a customer."""
10
11     def __init__(self):
12         """Initialize an empty shopping cart"""
13         self.cart_items=[]
14         self.total_price=0
15
16     def add_item(self,item,quantity):
17         """Add an item to the shopping cart with the given quantity"""
18
19         try:
20             if quantity < 1:
21                 raise ValueError("Quantity must be greater than or equal to 1")
22             self.cart_items.append((item,quantity))
23             self.total_price += item.price * quantity
24         except ValueError as ve:
25             print(f"Error : {ve}")
26
27     def remove_item(self,item,quantity):
28         """Remove an item from the shopping cart with the given quantity"""
29
30         for i, (cart_item, cart_quantity) in enumerate(self.cart_items):
31             if cart_item == item:
```

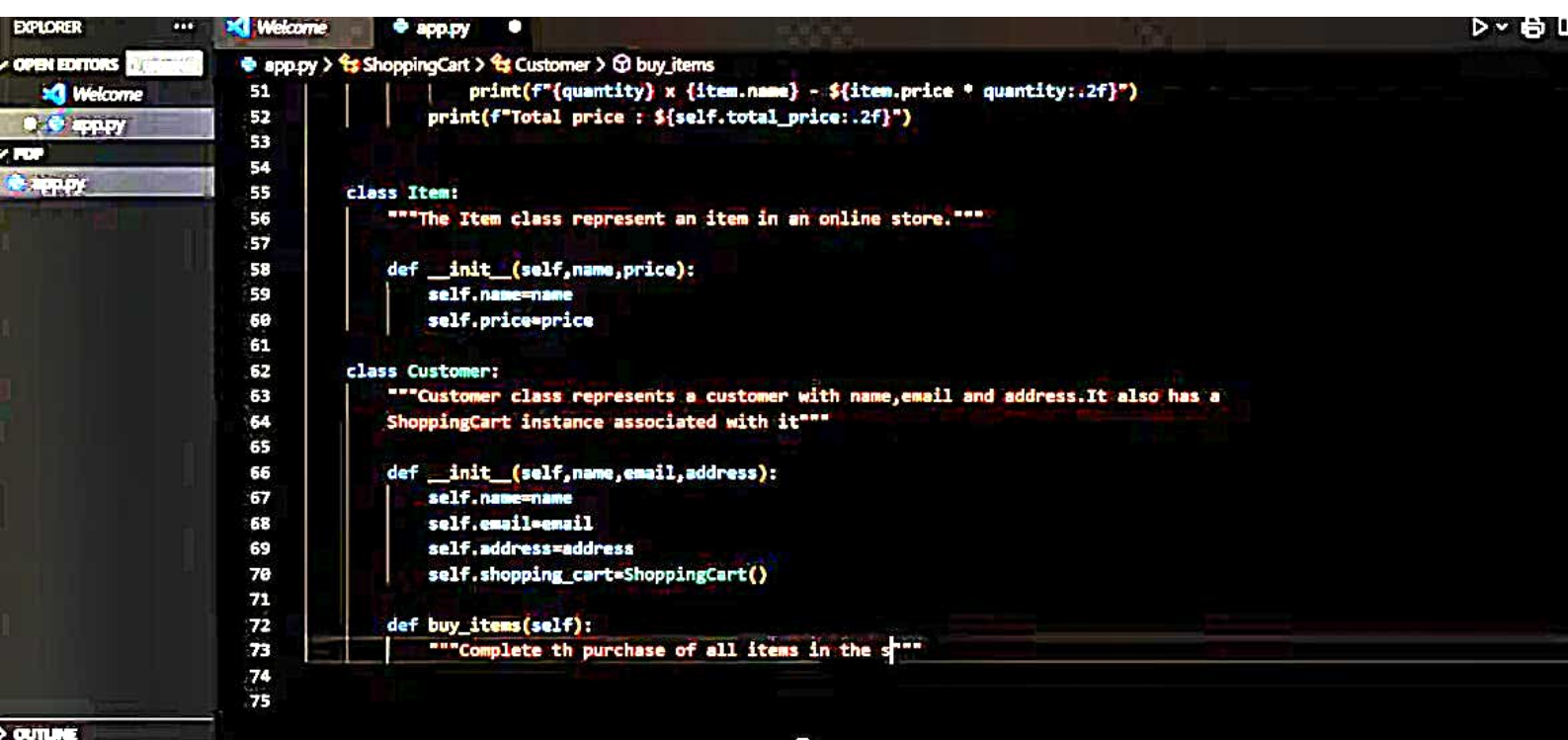
```
app.py > ShoppingCart > add_item
def remove_item(self, item, quantity):
    """Remove an item from the shopping cart with the given quantity"""
    for i, (cart_item, cart_quantity) in enumerate(self.cart_items):
        if cart_item == item:
            if cart_quantity > quantity:
                self.cart_items[i] = (cart_item, cart_quantity - quantity)
                self.total_price -= item.price * quantity
            elif cart_quantity == quantity:
                self.cart_items.pop(i)
                self.total_price -= item.price * quantity
            else:
                print(f"Error: Cannot remove {quantity} {item.name}(s), only {cart_quantity} available in cart")
                break

def view_cart(self):
    """View the current items in the shopping cart"""
    if not self.cart_items:
        print("Your Shopping Cart is empty.")
    else:
        print('Current Items in your shopping cart:')
        for item, quantity in self.cart_items:
            print(f"{quantity} x {item.name} - ${item.price * quantity:.2f}")
        print(f"Total price : ${self.total_price:.2f}")
```



```

51         print(f"{quantity} x {item.name} = ${item.price * quantity:.2f}")
52         print(f"Total price : ${self.total_price:.2f}")
53
54
55 class Item:
56     """The Item class represent an item in an online store."""
57
58     def __init__(self, name, price):
59         self.name = name
60         self.price = price
61
62 class Customer:
63     """Customer class represents a customer with name, email and address. It also has a
64     ShoppingCart instance associated with it"""
65
66     def __init__(self, name, email, address):
67         self.name = name
68         self.email = email
69         self.address = address
70         self.shopping_cart = ShoppingCart()
71
72
73
```


```
54
55
56     """The Item class represent an item in an online store."""
57
58     def __init__(self,name,price):
59         self.name=name
60         self.price=price
61
62 class Customer:
63     """Customer class represents a customer with name,email and address.It also has a
64     ShoppingCart instance associated with it"""
65
66     def __init__(self,name,email,address):
67         self.name=name
68         self.email=email
69         self.address=address
70         self.shopping_cart=ShoppingCart()
71
72     def buy_items(self):
73         """Complete the purchase of all items in the shopping cart."""
74
75         if not self.shopping_cart.cart_items:
76
```

```
EXPLORER
...
Welcome
app.py
app.py

app.py > ...

58 def __init__(self, name, price):
59     self.name = name
60     self.price = price
61
62 class Customer:
63     """Customer class represents a customer with name, email and address. It also has a
64     ShoppingCart instance associated with it"""
65
66     def __init__(self, name, email, address):
67         self.name = name
68         self.email = email
69         self.address = address
70         self.shopping_cart = ShoppingCart()
71
72     def buy(self, item):
73         """Buy an item"""
74         self.shopping_cart.add(item)
75
76     def __str__(self):
77         return f"Customer(name={self.name}, email={self.email}, address={self.address})"
78
79     def __repr__(self):
80         return f"Customer(name={self.name}, email={self.email}, address={self.address})"
81
```

```
app.py > ShoppingCart > Customer > buy_items
33:         self.cart_items[i]=(cart_item, cart_quantity-quantity)
34:         self.total_price -= item.price * quantity
35:
36:         elif cart_quantity == quantity:
37:             self.cart_items.pop(i)
38:             self.total_price -= item.price * quantity
39:         else:
40:             print(f"Error: Cannot remove {quantity} {item.name}(s), only {cart_quantity} available in cart")
41:             break
42:
43: def view_cart(self):
44:     """View the current items in the shopping cart"""
45:
46:     if not self.cart_items:
47:         print("Your Shopping Cart is empty.")
48:     else:
49:         print('Current Items in your shopping cart:')
50:         for item, quantity in self.cart_items:
51:             print(f"{quantity} x {item.name} - ${item.price * quantity:.2f}")
52:         print(f"Total price : ${self.total_price:.2f}")
53:
54:
55: class Item:
56:     """The Item class represent an item in an online store."""
57:
```

variables
ShoppingCart: None

Paused on step
app.py

app.py > Customer > _init_

47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68

```
print("your Shopping Cart is empty.")  
else:  
    print('Current Items in your shopping cart:')  
    for item,quantity in self.cart_items:  
        print(f"{quantity} x {item.name} - ${item.price * quantity:.2f}")  
    print(f"Total price : ${self.total_price:.2f}")  
  
class Item:  
    """The Item class represent an item in an online store."""  
    def __init__(self,name,price):  
        self.name=name  
        self.price=price  
  
class Customer:  
    """Customer class represents a customer with name,email and address.It also has a  
    ShoppingCart instance associated with it"""  
    def __init__(self,name,email,address):  
        self.name=name  
        self.email=email  
        self.address=address  
        self.shopping_cart=ShoppingCart()
```

Python Debug Console

+

⌵

🗑

⋮

➤

```
Locals
> special variables
(return) ShoppingCart: None
Global:
```

ALL STACK **FOUNDATIONS**

ALL STACK **FOUNDATIONS**

STUDIES

[illegible]

- ❑ Raised Exceptions
- ❑ Uncaught Exceptions

```

52 |         print(f"Total price : ${self.total_price:.2f}")
53 |
54 |
55 | class Item:
56 |     """The Item class represent an item in an online store."""
57 |     def __init__(self,name,price):
58 |         self.name=name
59 |         self.price=price
60 |
61 | class Customer:
62 |     """Customer class represents a customer with name,email and address.It also has a
63 |         ShoppingCart instance associated with it"""
64 |     def __init__(self,name,email,address):
65 |         self.name=name
66 |         self.email=email
67 |         self.address=address
68 |         self.shopping_cart=ShoppingCart()
69 |
70 |     def buy_items(self):
71 |         """Complete the purchase of all items in the shopping cart."""
72 |
73 |

```

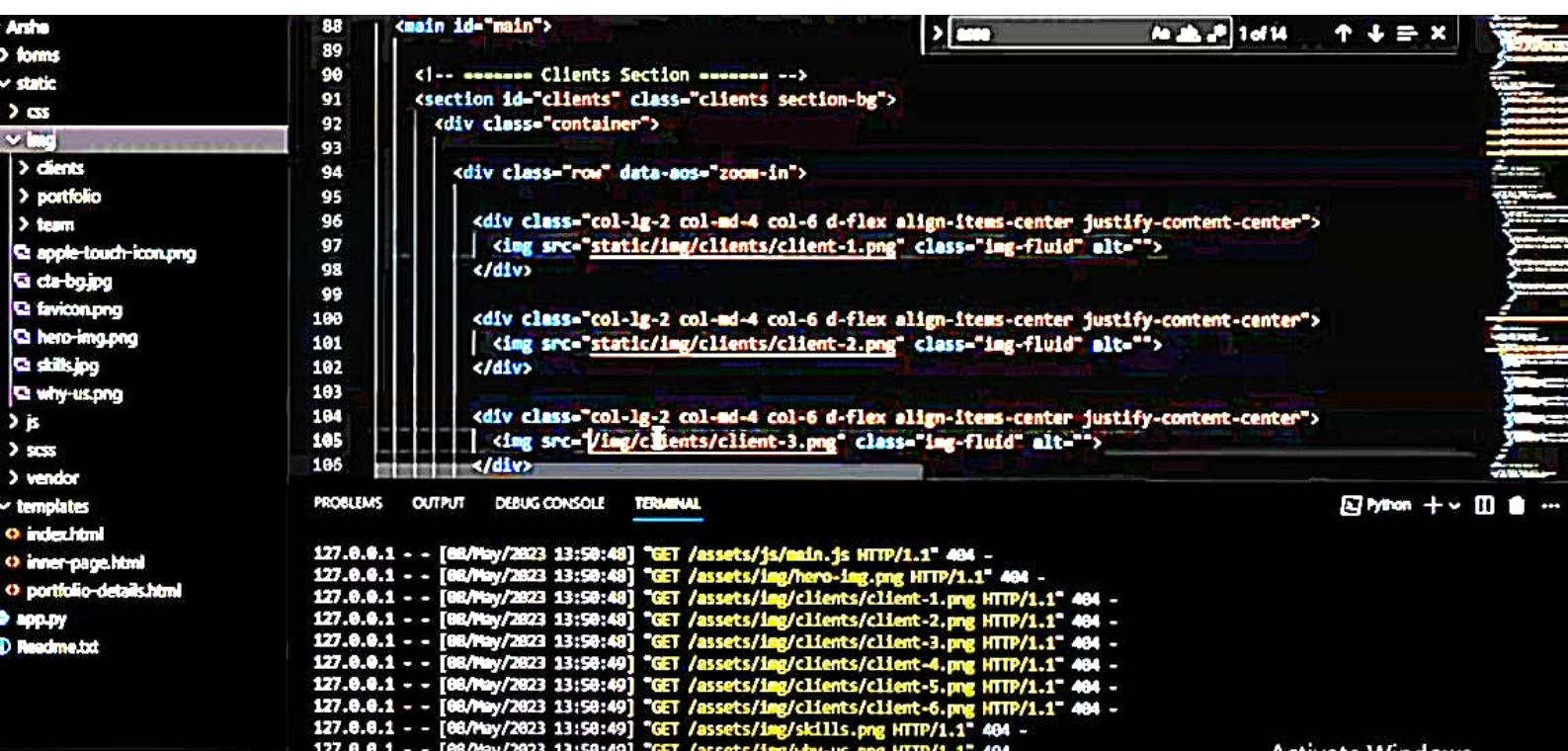
PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

Python Debug Console





SUPERSTORE

[Home](#)

[Dashboard](#)

[Story](#)

[Report](#)

[Contact](#)

[Get Started](#)

Sales Superstore project

We are team of talented designers making websites
with Bootstrap

[Get Started](#)



[Watch Video](#)



Appliances Binders Chairs Envelopes Furnishings Machines Phones Supplies

Region and Profit

Region	Profit
Central	39,706.36
East	91,622.78
South	46,749.43
West	108,418.45
Summary	286,397.02

Sales for State regions



Subscribe

ARSHA

A108 Adam Street
New York, NY 535022
United States

Phone: +1 5589 55488 55
Email: info@example.com

Useful Links

- > Home
- > About us
- > Services
- > Terms of service
- > Privacy policy

Our Services

- > Web Design
- > Web Development
- > Product Management
- > Marketing
- > Graphic Design

Our Social Networks

Cras fermentum odio eu feugiat lide
par naso tierra videa magna derita
valies



CONTACT

Magnam dolores commodi suscipit. Necessitatibus eius consequatur ex aliquid fuga eum quidem. Sit sint consectetur velit. Quisquam quos quisquam cupiditate. Et nemo qui impedit suscipit alias ea. Quia fugiat sit in iste officiis commodi quidem hic quas.



Location:

A108 Adam Street, New York, NY 535022



Email:

info@example.com



Call:

+1 5589 55488 55s

Your Name

Your Email

Subject

Message

Activate Windows

DASHBOARD

This is our sales dashboard.



Tab 1

Sales	Quantity	Profit	Product Name

Profit by Sub-Category colored by Category



Sales by Product Name colored by Category



Activate Windows
Go to Settings to activate Windows



DASHBOARD

This is our sales dashboard.



Tab 1



Profit by Sub-Category colored by Category

Sales by Product Name colored by Category

Activate Windows
Go to Settings to activate Windows.

Appliances Binders Chairs Envelopes Furnishings Machines Phones Supplies

Region and Profit

Region	Profit
Central	39,706.36
East	91,522.78
South	46,749.43
West	108,418.45
Summary	286,397.02

Sales for State regions



Quick access

- Desktop
- test_images
- Downloads
- Documents
- Pictures
- 2023-04-28 10.27.20 Shanawaz Anwar's Zoom
- Arsha
- garbage_classification
- memes_dataset
- OneDrive
- OneDrive
- This PC
- 3D Objects
- Desktop
- Documents
- Downloads
- Music

name	date modified	type	size
index	08-05-2023 14:06	Chrome HTML Do...	29 KB
inner-page	08-05-2023 11:23	Chrome HTML Do...	9 KB
portfolio-details	08-05-2023 11:23	Chrome HTML Do...	11 KB



Call To Action

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Call To Action

STORY

This is a visual narrative of our project.

SUPERSTORE

[Home](#)

[Dashboard](#)

[Story](#)

[Report](#)

[Contact](#)

[Get Started](#)

Sales Superstore project

We are team of talented designers making websites with Bootstrap

[Get Started](#)

[Watch Video](#)



`app.py = project_bootstrap = Visual Studio Code`

myob


BELIMO

LifeGroups

Lilly

citrus

Trustly
Activate windows

 download free |

Q download free - Google Search

Q download free fire

Q download free fire max

Q download free fire for pc

Q download free vpn

Q download free games for pc

Q download free html templates

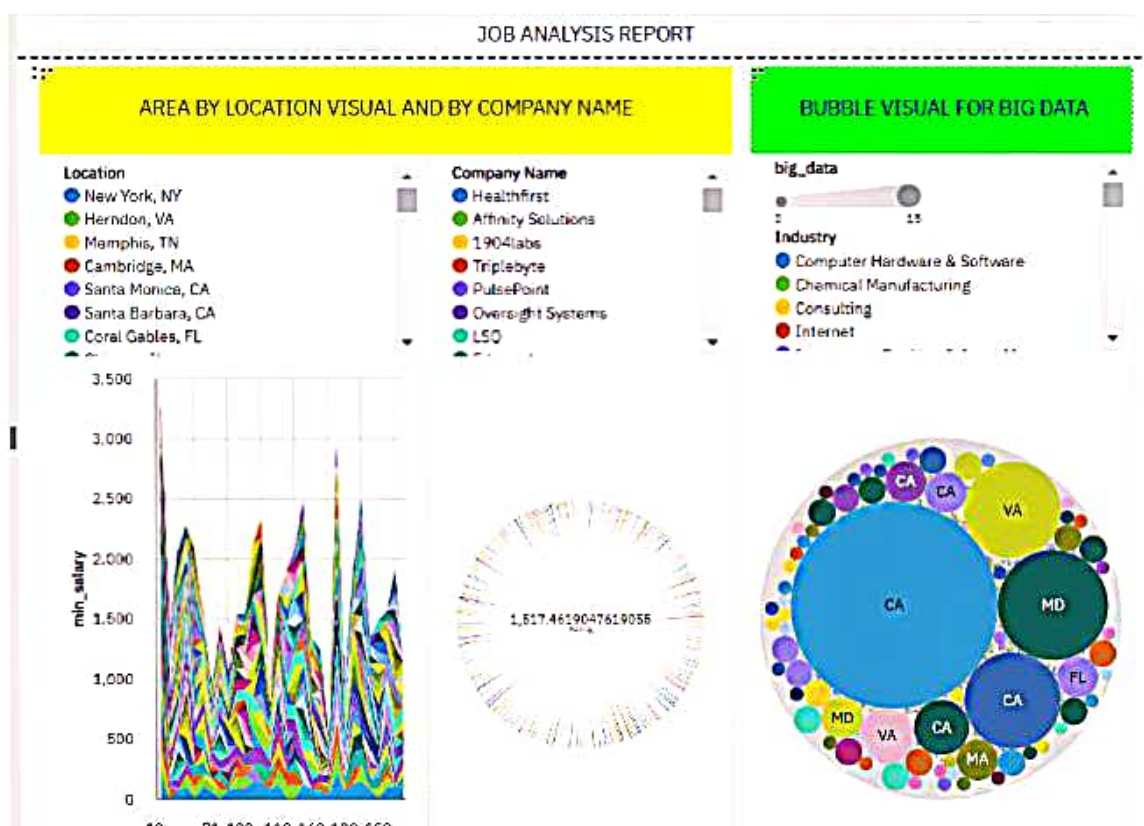
Q download free resume templates

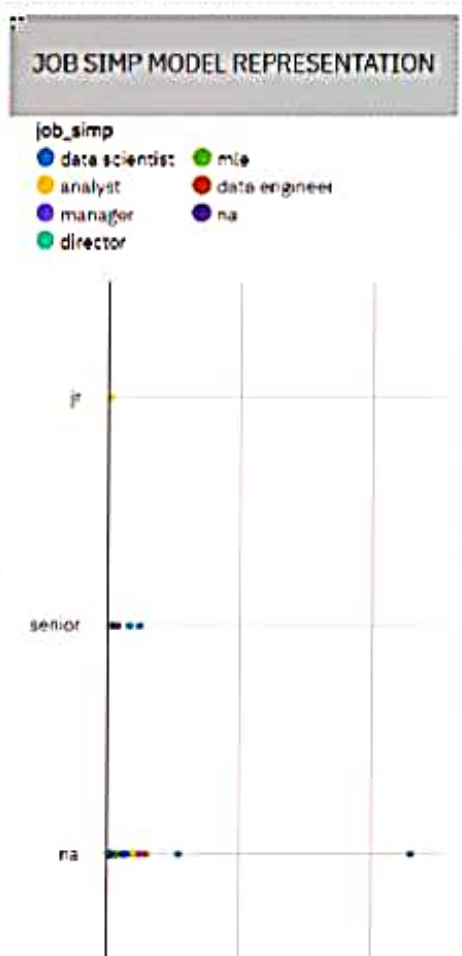
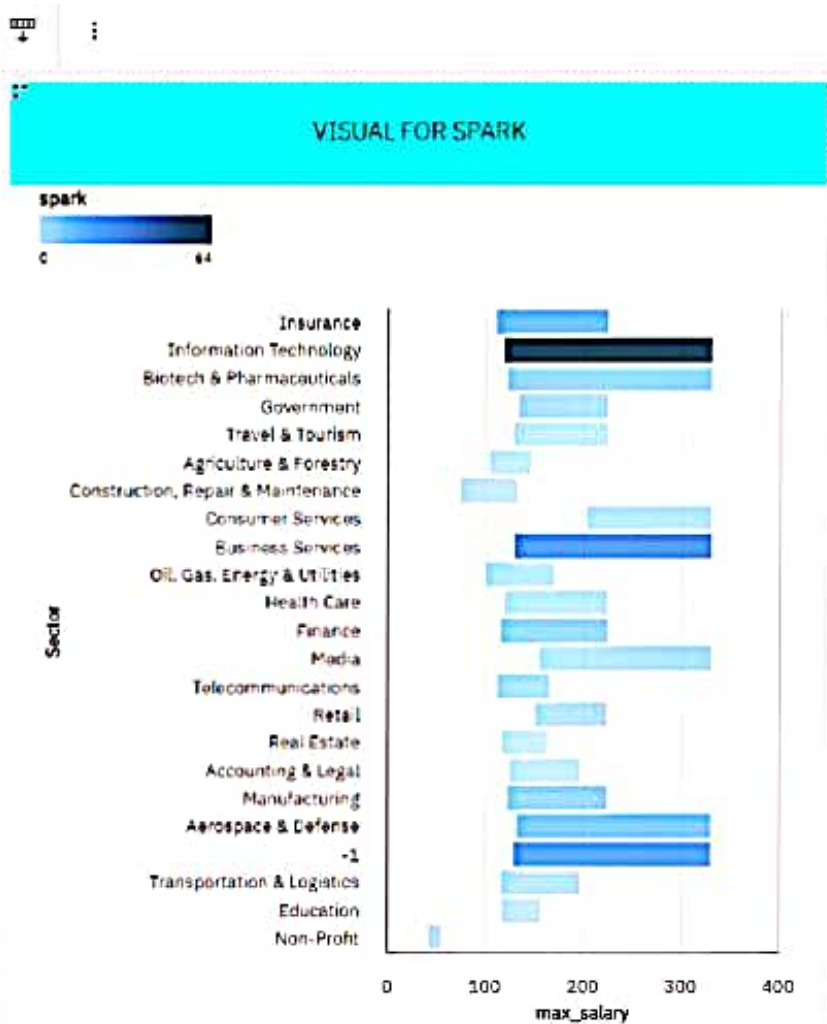
Q download free fire max for pc

Q download free images



nd
A NM DATA MODULE
Cleaned_DS_Jobs.csv

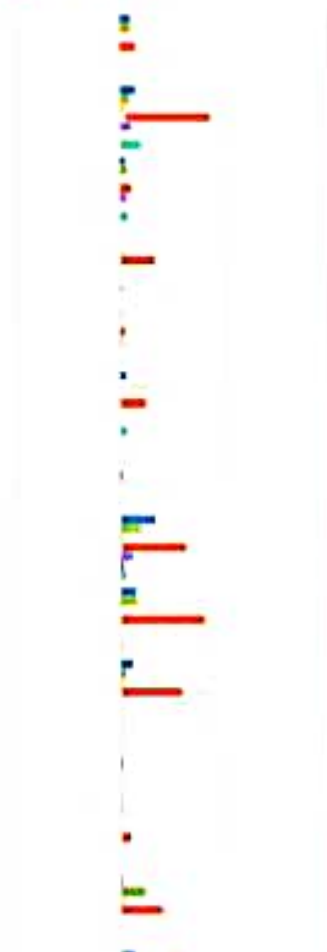




```

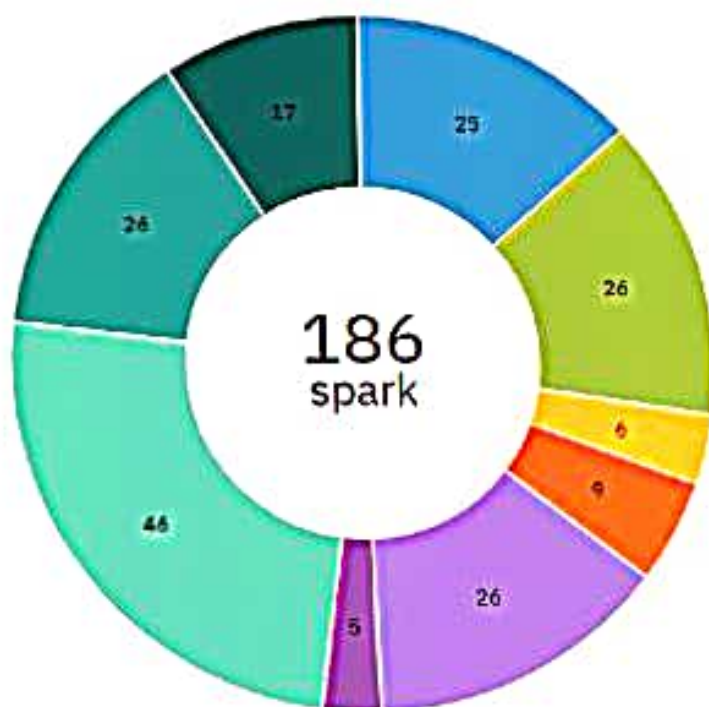
job_simp
analyst
na
manager
data scientist
mle
director
data engineer

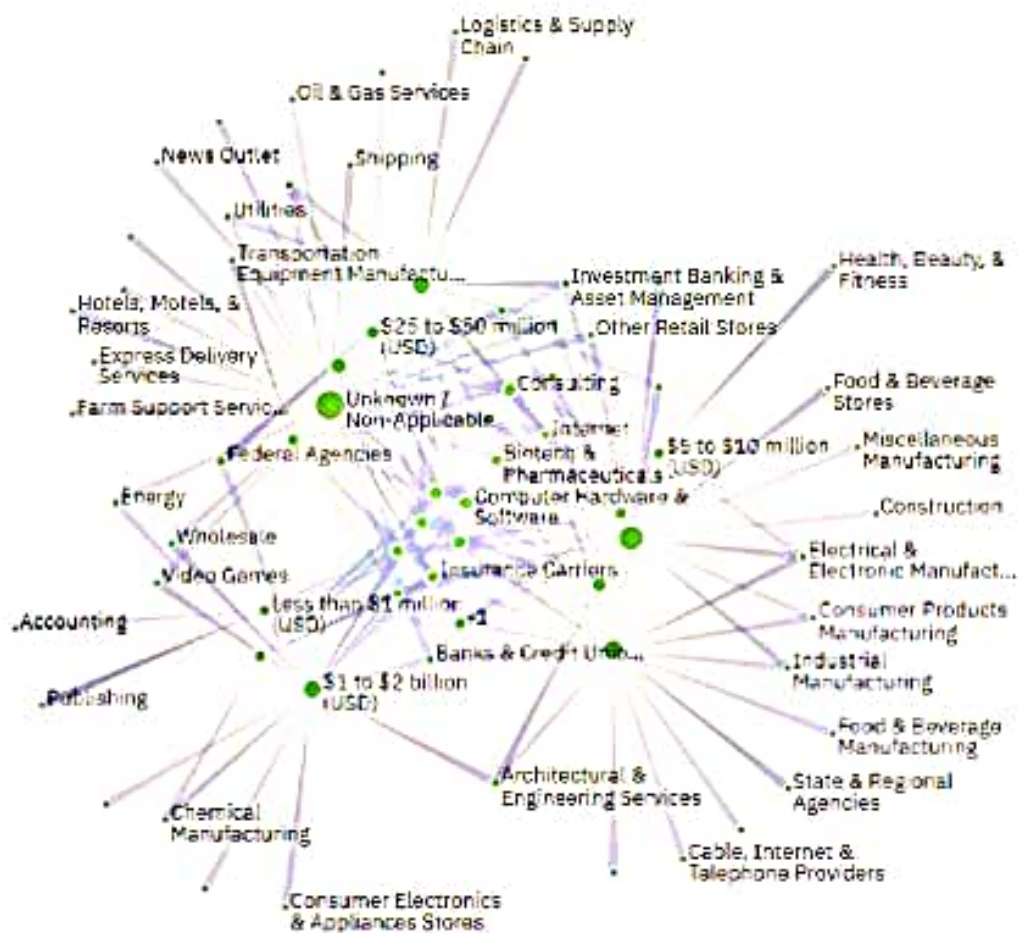
```



Size

- 10000+ employees
- 501 to 1000 employees
- 1 to 50 employees
- 51 to 200 employees
- 1001 to 5000 employees
- 5001 to 10000 employees
- 201 to 500 employees
- Unknown
- 1



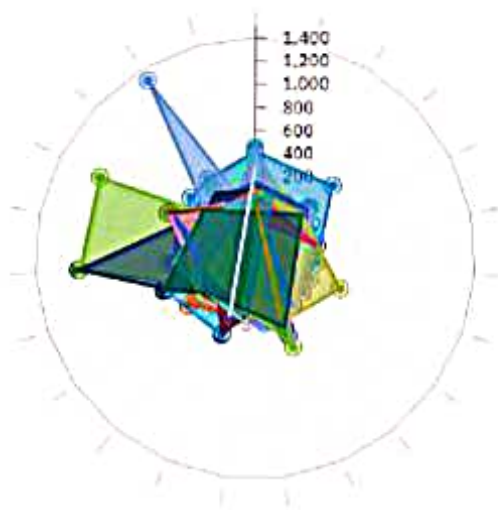


Company Name

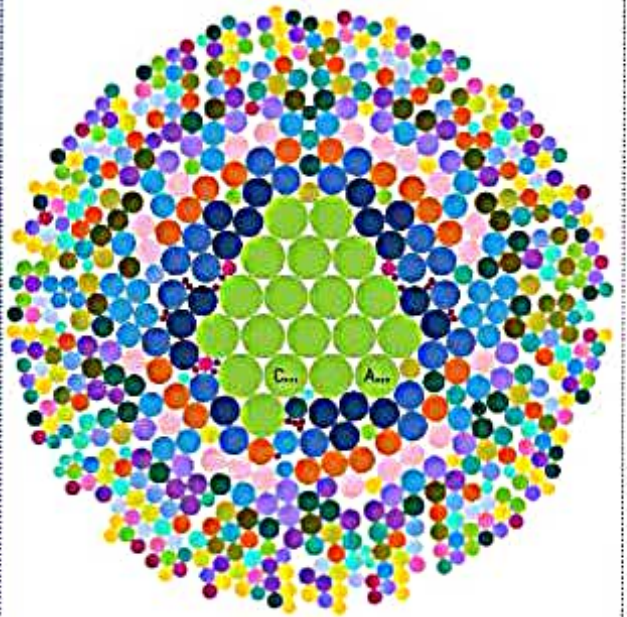
- Analysis Group
- PNNL
- Oversight Systems
- Great-Circle Technologies
- Dermologica
- Autodesk

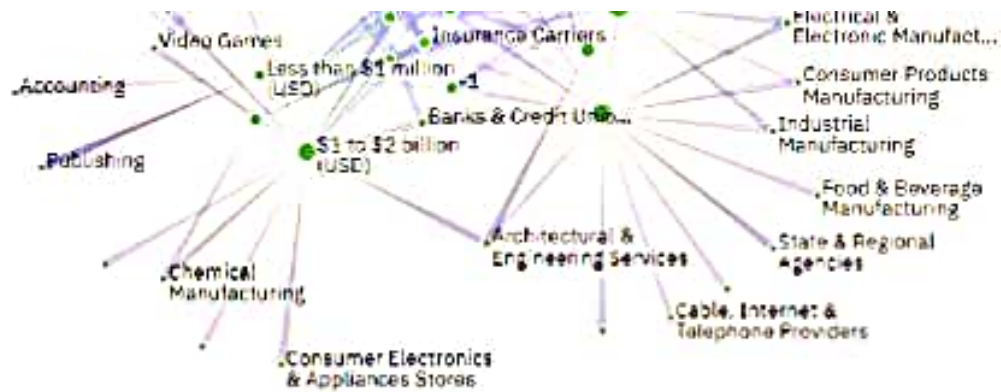
- HQ Insights
- Old World Industries
- LSQ
- Eversight
- Bayview Asset Management
- Comprehensive Healthcare

- XSELL Technologies
- Buckman
- Sandhills Global
- Postmates - Corporate HQ
- IZEA
- Evidation



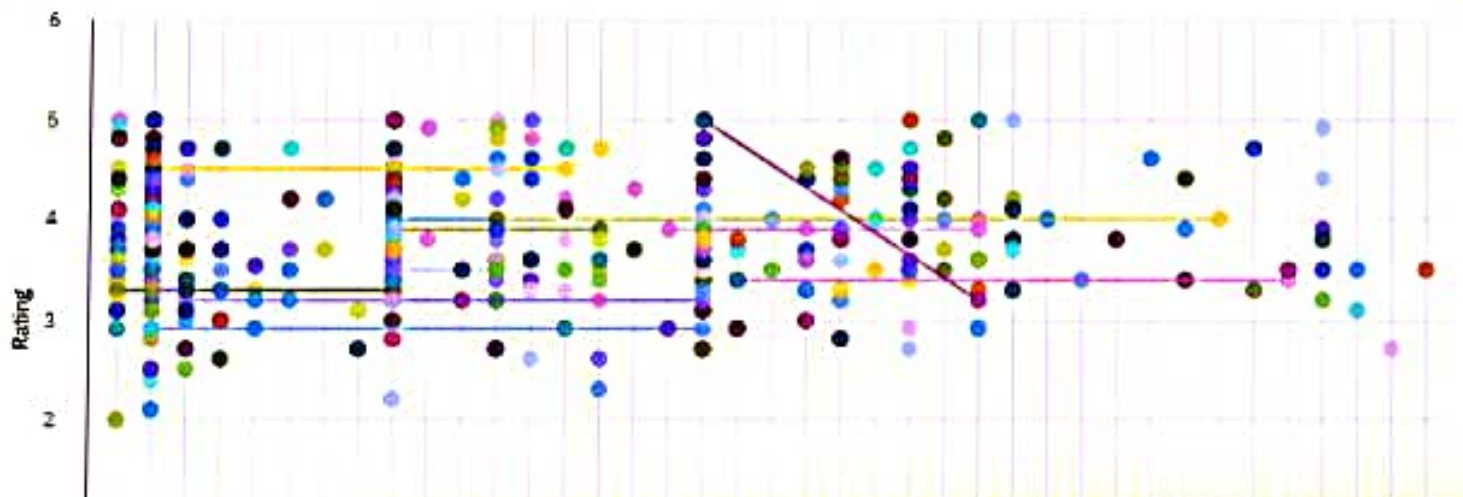
Salary Estimate

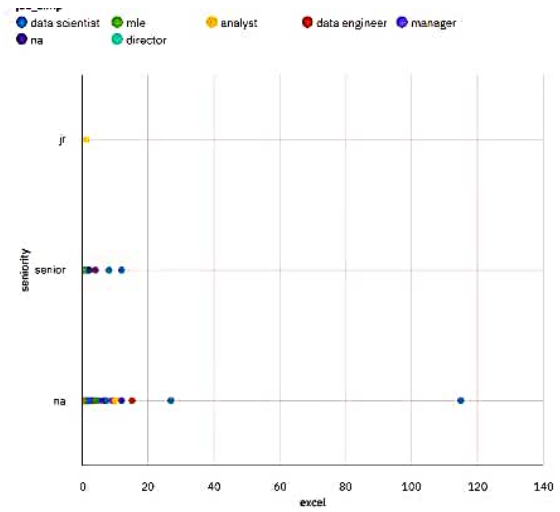
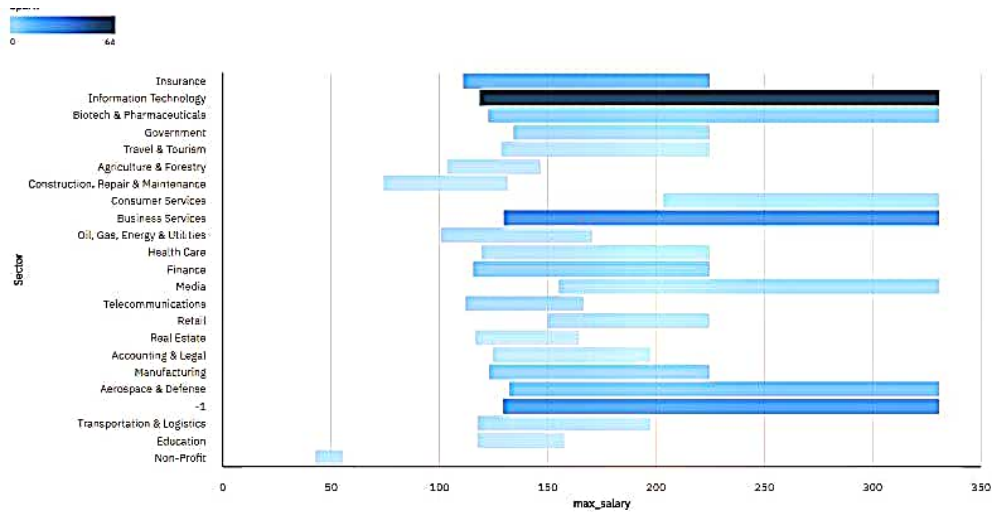


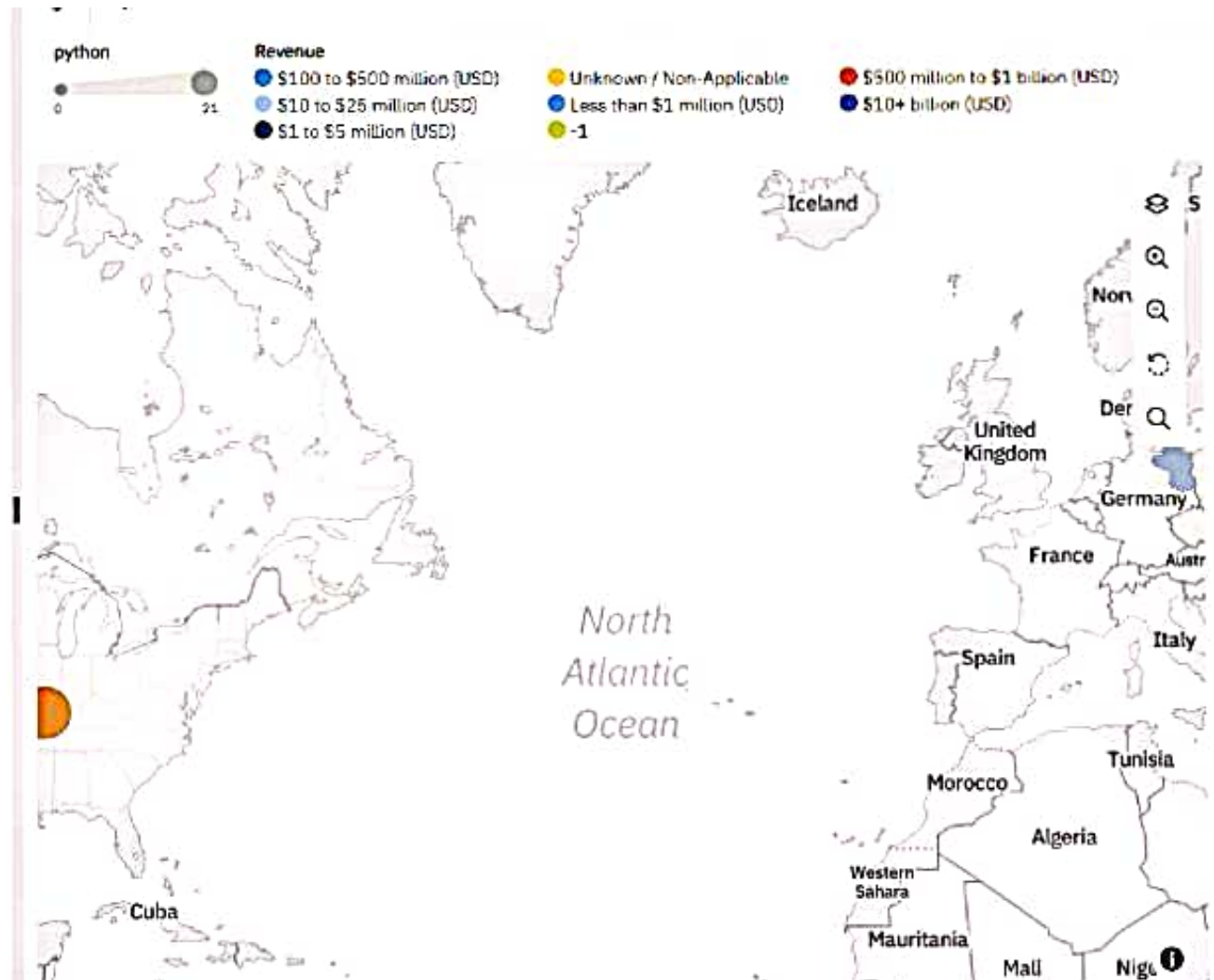


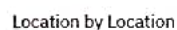
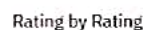
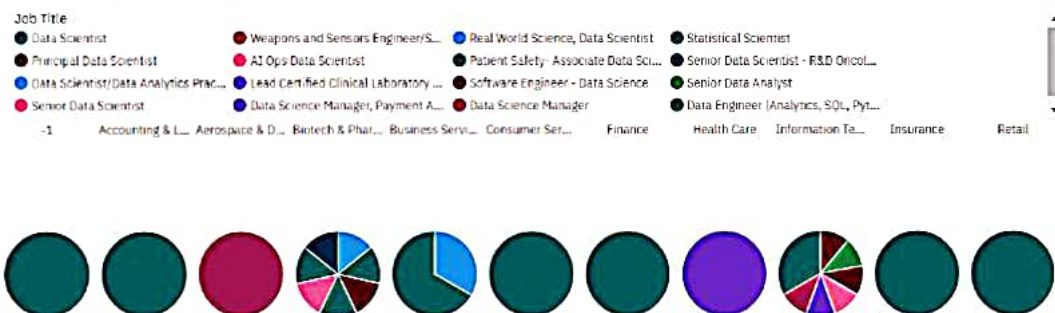
Company Name

- | | | |
|---------------------------|--------------------------|--------------------------|
| Analysis Group | HG Insights | XSELL Technologies |
| PNNL | Old World Industries | Buckman |
| Oversight Systems | LSQ | Sandhill Global |
| Great-Circle Technologies | Eversight | Postmates - Corporate HQ |
| Dermologica | Bayview Asset Management | IZEA |
| Autodesk | Comprehensive Healthcare | Evidation |
| Tecolote Research | Tivity Health | HP Inc. |











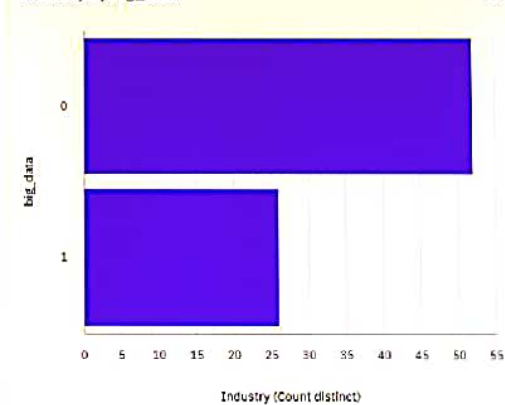
DATA ANALYSIS FOR JOBS

company_age by job_state and seniority

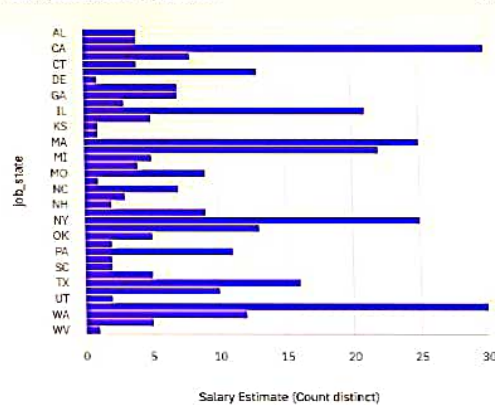


JOB OVERVIEW | **BAR CHARTS** | BAR GRAPHS | OTHERS | OTHERS

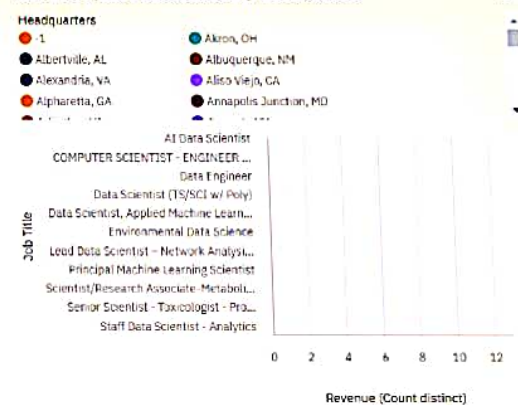
Industry by big_data



Salary Estimate by job_state



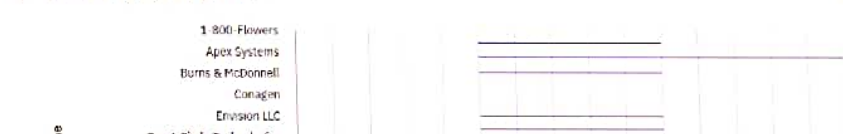
Revenue by Job Title colored by Headquarters

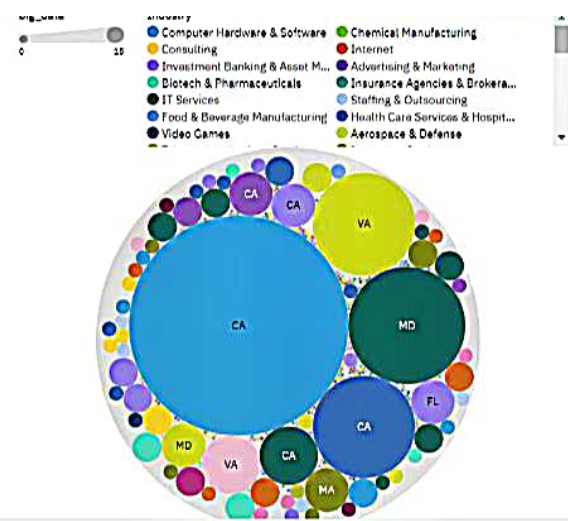
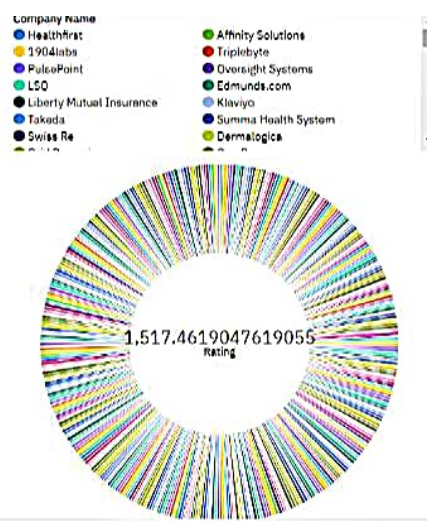
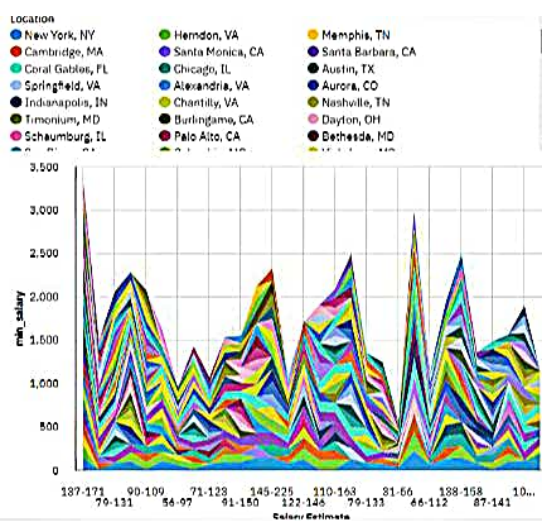


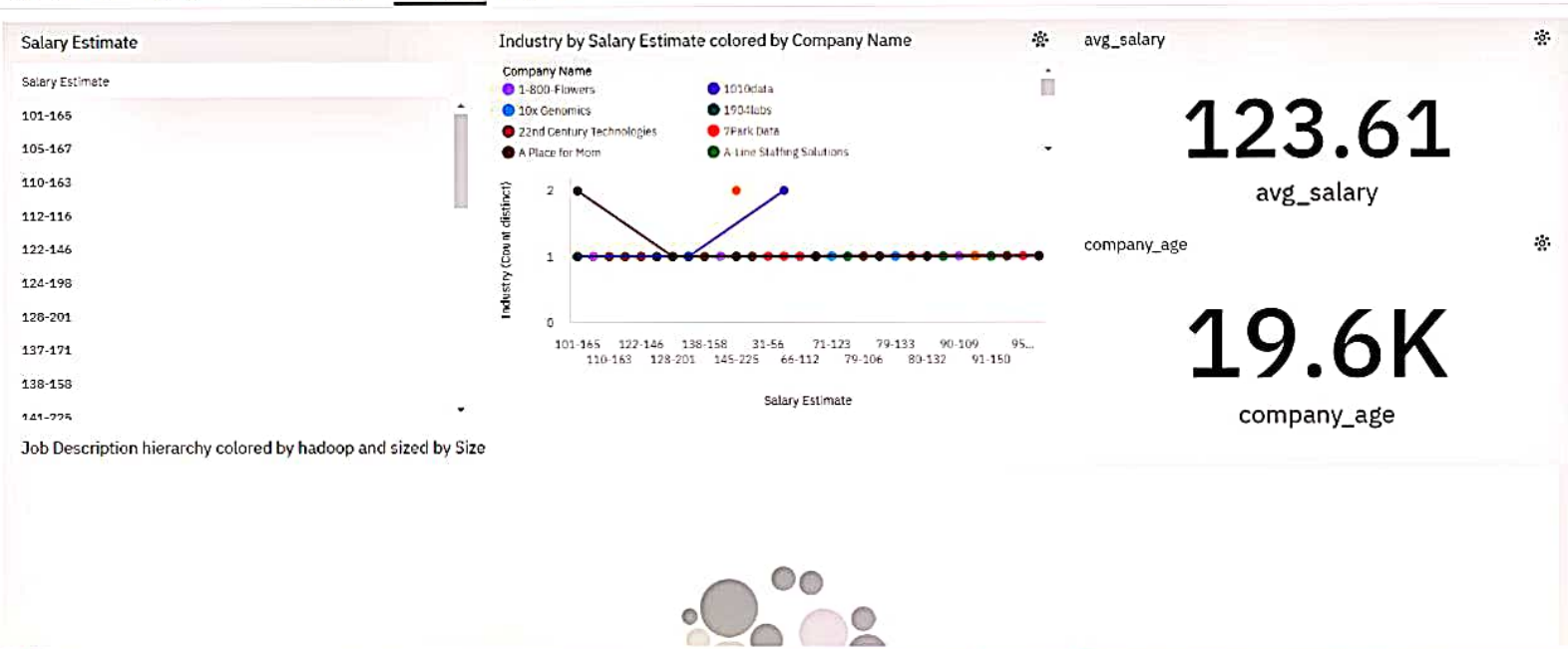
Rating by Headquarters colored by same_state

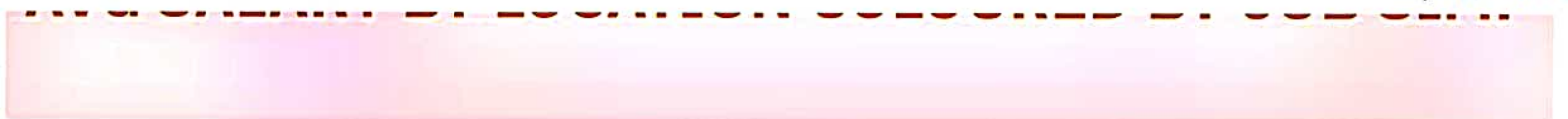


Location by Company Name









avg_salary by Location colored by job_simp

