supermarket

March 20, 2024

[4]:
```
import pandas as pd

# Define initial products and their quantities
initial_products = {
    'Product': ['Apple', 'Banana', 'Orange', 'Milk', 'Bread', 'Carrot',␣
 ↪'Tomato', 'Spinach', 'Chocolate', 'Cookies',
                 'Grapes', 'Broccoli', 'Potato', 'Onion', 'Rice', 'Chicken',␣
 ↪'Soap', 'Toilet Paper', 'Detergent',
                 'Pen', 'Notebook', 'Pencil', 'Eraser', 'Sharpener',␣
 ↪'Toothpaste', 'Shampoo', 'Conditioner', 'Towel', 'Cheese',
                 'Yogurt', 'Sugar', 'Salt', 'Biscuits', 'Tea', 'Coffee'],
    'Quantity': [200, 300, 240, 20, 100, 160, 180, 140, 50, 60, 120, 100, 150,␣
 ↪200, 80, 100, 150, 100, 80, 50, 60, 70, 80, 90, 100,80, 70, 50, 30, 40, 100,␣
 ↪120, 60, 90, 110],
    'Price': [1.50, 0.75, 2.00, 3.00, 1.50, 1.00, 0.75, 1.25, 2.50, 1.75, 2.50,␣
 ↪1.80, 0.50, 0.75, 4.00, 5.00, 2.00, 3.50, 6.00,
               1.00, 2.50, 0.50, 0.25, 0.75, 1.80, 4.50, 5.00, 8.00, 2.75, 1.50,␣
 ↪2.00, 1.00, 2.50, 3.00, 4.00],
    'Category': ['Fruit', 'Fruit', 'Fruit', 'Dairy', 'Bakery', 'Vegetable',␣
 ↪'Vegetable', 'Vegetable', 'Snack', 'Snack',
                  'Fruit', 'Vegetable', 'Vegetable', 'Vegetable', 'Grain',␣
 ↪'Meat', 'Home Essentials', 'Home Essentials', 'Home Essentials',
                  'Stationary', 'Stationary', 'Stationary', 'Stationary',␣
 ↪'Stationary', 'Personal Care', 'Personal Care', 'Personal Care', 'Home␣
 ↪Essentials', 'Dairy', 'Dairy', 'Groceries', 'Groceries', 'Snack',␣
 ↪'Groceries', 'Groceries'],
    'Unit': ['kg', 'kg', 'kg', 'liter', 'pound', 'kg', 'kg', 'kg', 'pack',␣
 ↪'pack',
              'kg', 'kg', 'kg', 'kg', 'kg', 'kg', 'pack', 'pack', 'pack', 'unit',
              'unit', 'unit', 'unit', 'unit', 'unit', 'liter', 'liter', 'unit',␣
 ↪'kg', 'pack', 'kg', 'kg', 'pack', 'kg', 'kg']
}

# Create DataFrame for products
df_products = pd.DataFrame(initial_products)

# Fixing price per kg for applicable items
```

```
df_products.loc[df_products['Product'].isin(['Apple', 'Banana', 'Orange',␣
 ↪'Grapes']), 'Price'] /= 1.5
df_products.loc[df_products['Product'].isin(['Potato', 'Onion']), 'Price'] /= 0.
 ↪5

# Define offers
offers = {
   import pandas as pd

# Define initial products and their quantities
initial_products = {
    'Product': ['Apple', 'Banana', 'Orange', 'Milk', 'Bread', 'Carrot',␣
 ↪'Tomato', 'Spinach', 'Chocolate', 'Cookies',
                'Grapes', 'Broccoli', 'Potato', 'Onion', 'Rice', 'Chicken',␣
 ↪'Soap', 'Toilet Paper', 'Detergent',
                'Pen', 'Notebook', 'Pencil', 'Eraser', 'Sharpener',␣
 ↪'Toothpaste', 'Shampoo', 'Conditioner', 'Towel', 'Cheese',
                'Yogurt', 'Sugar', 'Salt', 'Biscuits', 'Tea', 'Coffee'],
    'Quantity': [200, 300, 240, 20, 100, 160, 180, 140, 50, 60, 120, 100, 150,␣
 ↪200, 80, 100, 150, 100, 80, 50, 60, 70, 80, 90, 100,80, 70, 50, 30, 40, 100,␣
 ↪120, 60, 90, 110],
    'Price': [1.50, 0.75, 2.00, 3.00, 1.50, 1.00, 0.75, 1.25, 2.50, 1.75, 2.50,␣
 ↪1.80, 0.50, 0.75, 4.00, 5.00, 2.00, 3.50, 6.00,
              1.00, 2.50, 0.50, 0.25, 0.75, 1.80, 4.50, 5.00, 8.00, 2.75, 1.50,␣
 ↪2.00, 1.00, 2.50, 3.00, 4.00],
    'Category': ['Fruit', 'Fruit', 'Fruit', 'Dairy', 'Bakery', 'Vegetable',␣
 ↪'Vegetable', 'Vegetable', 'Snack', 'Snack',
                 'Fruit', 'Vegetable', 'Vegetable', 'Vegetable', 'Grain',␣
 ↪'Meat', 'Home Essentials', 'Home Essentials', 'Home Essentials',
                 'Stationary', 'Stationary', 'Stationary', 'Stationary',␣
 ↪'Stationary', 'Personal Care', 'Personal Care', 'Personal Care', 'Home␣
 ↪Essentials', 'Dairy', 'Dairy', 'Groceries', 'Groceries', 'Snack',␣
 ↪'Groceries', 'Groceries'],
    'Unit': ['kg', 'kg', 'kg', 'liter', 'pound', 'kg', 'kg', 'kg', 'pack',␣
 ↪'pack',
             'kg', 'kg', 'kg', 'kg', 'kg', 'kg', 'pack', 'pack', 'pack', 'unit',
             'unit', 'unit', 'unit', 'unit', 'unit', 'liter', 'liter', 'unit',␣
 ↪'kg', 'pack', 'kg', 'kg', 'pack', 'kg', 'kg']
}

# Create DataFrame for products
df_products = pd.DataFrame(initial_products)

# Fixing price per kg for applicable items
df_products.loc[df_products['Product'].isin(['Apple', 'Banana', 'Orange',␣
 ↪'Grapes']), 'Price'] /= 1.5
```

```
df_products.loc[df_products['Product'].isin(['Potato', 'Onion']), 'Price'] /= 0.
↪5

# Define offers
offers = {
    global df_products
    index = df_products[df_products['Product'] == product].index
    if len(index) == 0:
        print("Sorry, that product is not available.")
        return bill_df_purchased, bill_df_free, 0
    current_quantity = df_products.loc[index, 'Quantity'].values[0]
    if current_quantity < quantity:
        print("Sorry, there is not enough stock for your request.")
        return bill_df_purchased, bill_df_free, 0
    df_products.loc[index, 'Quantity'] -= quantity
    price_per_unit = df_products.loc[index, 'Price'].values[0]
    total_price = price_per_unit * quantity
    print(f"You have successfully bought {quantity} {df_products.loc[index,␣
↪'Unit'].values[0]} of {product} for ${total_price:.2f}.")

    # Apply offer if applicable
    bill_df_purchased, bill_df_free = apply_offer(product, quantity,␣
↪bill_df_purchased, bill_df_free)

    bill_df_purchased = pd.concat([bill_df_purchased, pd.DataFrame({'Product':␣
↪[product], 'Quantity': [quantity], 'Total Price': [total_price]})],␣
↪ignore_index=True)
    return bill_df_purchased, bill_df_free, total_price

def update_stock(product, quantity):
    """Update stock for a product."""
    global df_products
    index = df_products[df_products['Product'] == product].index
    if len(index) == 0:
        print("Sorry, that product is not available.")
        return
    df_products.loc[index, 'Quantity'] += quantity
    if df_products.loc[index, 'Quantity'].values[0] < 20:
        print(f"Warning: Stock for {product} is low. Please restock.")
    else:
        print(f"Stock for {product} has been updated by {quantity}.")

def simulate_purchase():
    """Simulate a purchase."""
    display_products()
    total_bill = 0
    bill_df_purchased = pd.DataFrame(columns=['Product', 'Quantity', 'Total␣
↪Price'])
    bill_df_free = pd.DataFrame(columns=['Product', 'Quantity', 'Total Price'])
    while True:
        product = input("Enter the product you want to buy (or type 'done' to␣
↪finish): ")
        if product.lower() == 'done':
            break
        if product not in df_products['Product'].tolist():
            print("Invalid product! Please choose a product from the list.")
            continue
        try:
            quantity = float(input(f"Enter the quantity of {product}: "))
        except ValueError:
            print("Invalid input! Quantity must be a number.")
            continue
        if quantity <= 0:
            print("Invalid input! Quantity must be greater than zero.")
            continue
        bill_df_purchased, bill_df_free, product_price = buy_product(product,␣
↪quantity, bill_df_purchased, bill_df_free)
        total_bill += product_price
    if not bill_df_purchased.empty:
        print("\nPurchased Items:")
        print(bill_df_purchased)
    else:
        print("No items purchased.")
    if not bill_df_free.empty:
        print("\nFree Items:")
        print(bill_df_free)
    if not bill_df_purchased.empty:
        print(f"\nTotal bill for purchased items: ${total_bill:.2f}")
    print()

def simulate_stock_update():
    """Simulate updating stock."""
    display_products()
    product = input("Enter the product you want to update: ")
    quantity = int(input(f"Enter the quantity you want to add ({df_products.
↪loc[df_products['Product'] == product, 'Unit'].values[0]}): "))
    update_stock(product, quantity)
    print()
    display_products()

def main():
    """Main function to run the supermarket simulation."""
    while True:
        print("1. Buy Product")
        print("2. Update Stock")
        print("3. Exit")
        choice = input("Enter your choice: ")
        if choice == '1':
            simulate_purchase()
        elif choice == '2':
            simulate_stock_update()
        elif choice == '3':
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Please enter a valid option.")
        print()

if __name__ == "__main__":
    main()
```

```
1. Buy Product
2. Update Stock
3. Exit

Enter your choice:  1

Available Products:
         Product  Quantity     Price         Category   Unit
0          Apple       200  1.000000            Fruit     kg
1         Banana       300  0.500000            Fruit     kg
2         Orange       240  1.333333            Fruit     kg
3           Milk        20  3.000000            Dairy  liter
4          Bread       100  1.500000           Bakery  pound
5         Carrot       160  1.000000        Vegetable     kg
6         Tomato       180  0.750000        Vegetable     kg
7        Spinach       140  1.250000        Vegetable     kg
8      Chocolate        50  2.500000            Snack   pack
9        Cookies        60  1.750000            Snack   pack
10        Grapes       120  1.666667            Fruit     kg
11      Broccoli       100  1.800000        Vegetable     kg
12        Potato       150  1.000000        Vegetable     kg
13         Onion       200  1.500000        Vegetable     kg
14          Rice        80  4.000000            Grain     kg
15       Chicken       100  5.000000             Meat     kg
16          Soap       150  2.000000  Home Essentials   pack
17  Toilet Paper       100  3.500000  Home Essentials   pack
18     Detergent        80  6.000000  Home Essentials   pack
19           Pen        50  1.000000       Stationary   unit
20      Notebook        60  2.500000       Stationary   unit
21        Pencil        70  0.500000       Stationary   unit
```

```
22        Eraser        80  0.250000       Stationary   unit
23     Sharpener        90  0.750000       Stationary   unit
24    Toothpaste       100  1.800000    Personal Care   unit
25       Shampoo        80  4.500000    Personal Care  liter
26   Conditioner        70  5.000000    Personal Care  liter
27         Towel        50  8.000000  Home Essentials   unit
28        Cheese        30  2.750000            Dairy     kg
29        Yogurt        40  1.500000            Dairy   pack
30         Sugar       100  2.000000        Groceries     kg
31          Salt       120  1.000000        Groceries     kg
32      Biscuits        60  2.500000            Snack   pack
33           Tea        90  3.000000        Groceries     kg
34        Coffee       110  4.000000        Groceries     kg

Enter the product you want to buy (or type 'done' to finish):  Notebook
Enter the quantity of Notebook:  2

You have successfully bought 2.0 unit of Notebook for $5.00.

Enter the product you want to buy (or type 'done' to finish):  Cheese
Enter the quantity of Cheese:  0.25

You have successfully bought 0.25 kg of Cheese for $0.69.

Enter the product you want to buy (or type 'done' to finish):  Chocolate
Enter the quantity of Chocolate:  3

You have successfully bought 3.0 pack of Chocolate for $7.50.

Enter the product you want to buy (or type 'done' to finish):  Toothpaste
Enter the quantity of Toothpaste:  2

You have successfully bought 2.0 unit of Toothpaste for $3.60.
Offer: Added 2.0 free Soap(s) with your purchase of 2.0 Toothpaste(s).

Enter the product you want to buy (or type 'done' to finish):  Shampoo
Enter the quantity of Shampoo:  0.5

You have successfully bought 0.5 liter of Shampoo for $2.25.

Enter the product you want to buy (or type 'done' to finish):  done


Purchased Items:
      Product  Quantity  Total Price
0    Notebook      2.00       5.0000
1      Cheese      0.25       0.6875
2   Chocolate      3.00       7.5000
3  Toothpaste      2.00       3.6000
4     Shampoo      0.50       2.2500

Free Items:
  Product  Quantity Total Price
```

```
0    Soap       2.0          0

Total bill for purchased items: $19.04


1. Buy Product
2. Update Stock
3. Exit

Enter your choice:  2

Available Products:
          Product  Quantity     Price         Category   Unit
0           Apple    200.00  1.000000            Fruit     kg
1          Banana    300.00  0.500000            Fruit     kg
2          Orange    240.00  1.333333            Fruit     kg
3            Milk     20.00  3.000000            Dairy  liter
4           Bread    100.00  1.500000           Bakery  pound
5          Carrot    160.00  1.000000        Vegetable     kg
6          Tomato    180.00  0.750000        Vegetable     kg
7         Spinach    140.00  1.250000        Vegetable     kg
8       Chocolate     47.00  2.500000            Snack   pack
9         Cookies     60.00  1.750000            Snack   pack
10         Grapes    120.00  1.666667            Fruit     kg
11       Broccoli    100.00  1.800000        Vegetable     kg
12         Potato    150.00  1.000000        Vegetable     kg
13          Onion    200.00  1.500000        Vegetable     kg
14           Rice     80.00  4.000000            Grain     kg
15        Chicken    100.00  5.000000             Meat     kg
16           Soap    149.00  2.000000  Home Essentials   pack
17   Toilet Paper    100.00  3.500000  Home Essentials   pack
18      Detergent     80.00  6.000000  Home Essentials   pack
19            Pen     50.00  1.000000       Stationary   unit
20       Notebook     58.00  2.500000       Stationary   unit
21         Pencil     70.00  0.500000       Stationary   unit
22         Eraser     80.00  0.250000       Stationary   unit
23      Sharpener     90.00  0.750000       Stationary   unit
24     Toothpaste     98.00  1.800000    Personal Care   unit
25        Shampoo     79.50  4.500000    Personal Care  liter
26    Conditioner     70.00  5.000000    Personal Care  liter
27          Towel     50.00  8.000000  Home Essentials   unit
28         Cheese     29.75  2.750000            Dairy     kg
29         Yogurt     40.00  1.500000            Dairy   pack
30          Sugar    100.00  2.000000        Groceries     kg
31           Salt    120.00  1.000000        Groceries     kg
32       Biscuits     60.00  2.500000            Snack   pack
33            Tea     90.00  3.000000        Groceries     kg
34         Coffee    110.00  4.000000        Groceries     kg
```

```
Enter the product you want to update:  Milk
Enter the quantity you want to add (liter):  50

Stock for Milk has been updated by 50.

Available Products:
         Product  Quantity     Price         Category   Unit
0          Apple    200.00  1.000000            Fruit     kg
1         Banana    300.00  0.500000            Fruit     kg
2         Orange    240.00  1.333333            Fruit     kg
3           Milk     70.00  3.000000            Dairy  liter
4          Bread    100.00  1.500000           Bakery  pound
5         Carrot    160.00  1.000000        Vegetable     kg
6         Tomato    180.00  0.750000        Vegetable     kg
7        Spinach    140.00  1.250000        Vegetable     kg
8      Chocolate     47.00  2.500000            Snack   pack
9        Cookies     60.00  1.750000            Snack   pack
10        Grapes    120.00  1.666667            Fruit     kg
11      Broccoli    100.00  1.800000        Vegetable     kg
12        Potato    150.00  1.000000        Vegetable     kg
13         Onion    200.00  1.500000        Vegetable     kg
14          Rice     80.00  4.000000            Grain     kg
15       Chicken    100.00  5.000000             Meat     kg
16          Soap    149.00  2.000000  Home Essentials   pack
17  Toilet Paper    100.00  3.500000  Home Essentials   pack
18     Detergent     80.00  6.000000  Home Essentials   pack
19           Pen     50.00  1.000000       Stationary   unit
20      Notebook     58.00  2.500000       Stationary   unit
21        Pencil     70.00  0.500000       Stationary   unit
22        Eraser     80.00  0.250000       Stationary   unit
23     Sharpener     90.00  0.750000       Stationary   unit
24    Toothpaste     98.00  1.800000    Personal Care   unit
25       Shampoo     79.50  4.500000    Personal Care  liter
26   Conditioner     70.00  5.000000    Personal Care  liter
27         Towel     50.00  8.000000  Home Essentials   unit
28        Cheese     29.75  2.750000            Dairy     kg
29        Yogurt     40.00  1.500000            Dairy   pack
30         Sugar    100.00  2.000000        Groceries     kg
31          Salt    120.00  1.000000        Groceries     kg
32      Biscuits     60.00  2.500000            Snack   pack
33           Tea     90.00  3.000000        Groceries     kg
34        Coffee    110.00  4.000000        Groceries     kg


1. Buy Product
2. Update Stock
3. Exit

Enter your choice:  3
```

```
Exiting program.
```