```
# prompt: add in code so that each store's total sales is listed at the end, as well as the total corporation sales, rounding all prices to tl

import random
from datetime import datetime, timedelta


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

    def __str__(self):
        return f"{self.name} (${self.price})"


class Order:
    def __init__(self, order_id, products, customer_id, store_id):
        self.order_id = order_id
        self.products = products
        self.customer_id = customer_id
        self.store_id = store_id
        self.timestamp = datetime.now() - timedelta(days=random.randint(0, 6),
                                                    hours=random.randint(0, 23),
                                                    minutes=random.randint(0, 59))

    def total_price(self):
        return sum(product.price for product in self.products)


class Customer:
    def __init__(self, customer_id):
        self.customer_id = customer_id
        self.orders = []

    def create_order(self, products_list, store_id):
        order_id = random.randint(1000, 9999)
        num_products = random.randint(1, len(products_list))
        selected_products = random.sample(products_list, num_products)
        selected_products.sort(key=lambda x: x.price, reverse=True)
        order = Order(order_id, selected_products, self.customer_id, store_id)
        self.orders.append(order)
        return order


class Store:
    def __init__(self, store_id):
        self.store_id = store_id
        self.orders = []

    def record_order(self, customer, products):
        order = customer.create_order(products, self.store_id)
        self.orders.append(order)
        return order

    def get_total_sales(self):
        return round(sum(order.total_price() for order in self.orders), 2)


class Corporation:
    def __init__(self, name):
        self.name = name
        self.stores = []

    def add_store(self, store):
        self.stores.append(store)

    def generate_sales_report(self):
        all_orders = []
        for store in self.stores:
            for order in store.orders:
                date_str = order.timestamp.strftime("%Y-%m-%d")
                time_str = order.timestamp.strftime("%H:%M:%S")
                product_info = [str(product) for product in order.products]
                all_orders.append((order.timestamp, date_str, time_str,
                                   order.store_id, order.customer_id,
                                   order.order_id, product_info,
                                   round(order.total_price(), 2)))

        all_orders.sort(key=lambda x: x[0], reverse=True)

        print("Date, Time, StoreID, CustomerID, OrderID, Products, Total")
        for timestamp, date_str, time_str, store_id, customer_id, order_id, products, total in all_orders:
            print(f"{date_str}, {time_str}, {store_id}, {customer_id}, {order_id}, {products}, {total}")

        # Calculate and print total sales for each store
        for store in self.stores:
            print(f"Store {store.store_id} Total Sales: ${store.get_total_sales()}")

        # Calculate and print total corporation sales
        total_corp_sales = round(sum(store.get_total_sales() for store in self.stores), 2)
        print(f"Total Corporation Sales: ${total_corp_sales}")


if __name__ == "__main__":
    corp = Corporation("Global Corp")
    for i in range(1, 11):
        corp.add_store(Store(i))

    # Top grocery brands and items
    top_15_grocery_brands = [
        "Whole Foods", "Trader Joe's", "Kroger", "Walmart", "Costco", "Aldi", "Safeway",
        "Publix", "Sprouts", "Albertsons", "Hy-Vee", "Meijer", "H-E-B", "Wegmans", "Giant"
    ]

    top_15_grocery_items_with_pricing = [
        ("Bread", 2.50),
        ("Milk", 3.00),
        ("Eggs", 2.00),
        ("Cheese", 5.00),
        ("Chicken Breast", 8.00),
        ("Ground Beef", 6.00),
        ("Apples", 1.50),
        ("Bananas", 0.50),
        ("Potatoes", 3.00),
        ("Rice", 2.00),
        ("Pasta", 1.50),
        ("Cereal", 4.00),
        ("Orange Juice", 3.50),
        ("Yogurt", 1.00),
        ("Butter", 2.50)
    ]

    generated_products = []
    for _ in range(20):  # Generate 20 random products
        brand = random.choice(top_15_grocery_brands)
        item, base_price = random.choice(top_15_grocery_items_with_pricing)
        # Introduce slight price variation based on brand
        price_variation = random.uniform(-0.1, 0.1)  # +/- 10% variation
        price = round(base_price * (1 + price_variation), 2)
        product_name = f"{brand} {item}"
        generated_products.append(Product(product_name, price))


    customers = [Customer(i) for i in range(1001, 1015)]

    for i in range(len(corp.stores)):
        corp.stores[i].record_order(customers[i % len(customers)], generated_products)

    corp.generate_sales_report()
```

```
Date, Time, StoreID, CustomerID, OrderID, Products, Total
2024-11-25, 20:01:15, 1, 1001, 7781, ['Walmart Cheese ($4.98)', 'H-E-B Cheese ($4.86)', 'Giant Cheese ($4.75)', 'Meijer Cereal ($3.64)',
2024-11-25, 06:04:15, 4, 1004, 4581, ['Giant Cheese ($4.75)', 'Meijer Bread ($2.54)', 'Publix Eggs ($2.01)', 'Meijer Apples ($1.53)'], 1
2024-11-24, 04:44:15, 8, 1008, 2717, ['Aldi Chicken Breast ($8.8)', 'Meijer Chicken Breast ($7.75)', 'H-E-B Cheese ($4.86)', 'Giant Chee
2024-11-24, 03:08:15, 10, 1010, 7444, ['Aldi Chicken Breast ($8.8)', 'Meijer Chicken Breast ($7.75)', 'Giant Cheese ($4.75)', 'Meijer Ce
2024-11-23, 22:54:15, 9, 1009, 8954, ['Aldi Chicken Breast ($8.8)', 'Walmart Cheese ($4.98)', 'H-E-B Cheese ($4.86)', 'Giant Cheese ($4.
2024-11-23, 18:20:15, 3, 1003, 4965, ['Aldi Chicken Breast ($8.8)', 'Meijer Chicken Breast ($7.75)', 'Walmart Cheese ($4.98)', 'Giant Ch
2024-11-23, 16:29:15, 2, 1002, 4868, ['Aldi Chicken Breast ($8.8)', 'Giant Cheese ($4.75)', 'Safeway Orange Juice ($3.35)', 'Costco Pota
2024-11-22, 12:27:15, 6, 1006, 9216, ['Walmart Cheese ($4.98)', 'H-E-B Cheese ($4.86)', 'Meijer Cereal ($3.64)', 'Costco Potatoes ($3.29
2024-11-20, 02:03:15, 5, 1005, 7436, ['Costco Potatoes ($3.15)', 'Meijer Bread ($2.54)', 'Aldi Bread ($2.34)', 'Publix Eggs ($2.01)'], 1
2024-11-19, 05:22:15, 7, 1007, 1247, ['Giant Cheese ($4.75)', 'Costco Potatoes ($3.29)', 'Costco Potatoes ($3.15)', 'Costco Bread ($2.72
Store 1 Total Sales: $35.17
Store 2 Total Sales: $39.91
Store 3 Total Sales: $46.17
Store 4 Total Sales: $10.83
Store 5 Total Sales: $10.04
Store 6 Total Sales: $25.84
Store 7 Total Sales: $22.93
Store 8 Total Sales: $45.71
Store 9 Total Sales: $39.44
Store 10 Total Sales: $40.37
Total Corporation Sales: $316.41
```

```
import pandas as pd
import random
from datetime import datetime, timedelta

# Define the Product class
class Product:
    def __init__(self, name, price):
        self.name = name
        self.price = price

# Define the Order class
class Order:
    def __init__(self, order_id, products, customer_id, store_id):
        self.order_id = order_id
        self.products = products
        self.customer_id = customer_id
        self.store_id = store_id
        self.timestamp = datetime.now() - timedelta(days=random.randint(0, 6),
                                                    hours=random.randint(0, 23),
                                                    minutes=random.randint(0, 59))

    def total_price(self):
        return sum(product.price for product in self.products)

# Define the Customer class
class Customer:
    def __init__(self, customer_id):
        self.customer_id = customer_id

    def create_order(self, product_list, store_id):
        order_id = random.randint(1000, 9999)
        selected_products = random.sample(product_list, random.randint(1, len(product_list)))
        return Order(order_id, selected_products, self.customer_id, store_id)

# Define the Store class
class Store:
    def __init__(self, store_id):
        self.store_id = store_id
        self.orders = []

    def add_order(self, order):
        self.orders.append(order)

# Define the Corporation class to manage data generation and reporting
class Corporation:
    def __init__(self, name):
        self.name = name
        self.stores = []
        self.customers = []

    def add_store(self, store):
        self.stores.append(store)

    def add_customer(self, customer):
        self.customers.append(customer)

    def generate_sales_data(self, products):
        for store in self.stores:
            for customer in self.customers:
                order = customer.create_order(products, store.store_id)
                store.add_order(order)

    def generate_sales_report(self):
        all_orders = []
        for store in self.stores:
            for order in store.orders:
                all_orders.append({
                    "Timestamp": order.timestamp,
                    "Date": order.timestamp.strftime("%Y-%m-%d"),
                    "Time": order.timestamp.strftime("%H:%M:%S"),
                    "StoreID": order.store_id,
                    "CustomerID": order.customer_id,
                    "OrderID": order.order_id,
                    "Products": [product.name for product in order.products],
                    "TotalPrice": round(order.total_price(), 2)
                })
        return pd.DataFrame(all_orders)

# Set up the corporation and generate data
corp = Corporation("Global Corp")

# Add 10 stores
for i in range(1, 11):
    corp.add_store(Store(i))

# Add 14 customers
for i in range(1001, 1015):
    corp.add_customer(Customer(i))

# Define a list of products with prices
products = [
    Product("Apple", 1.50), Product("Banana", 0.75), Product("Orange", 1.00),
    Product("Grapes", 2.50), Product("Strawberry", 3.00), Product("Broccoli", 1.75),
    Product("Carrot", 0.50), Product("Tomato", 1.25), Product("Potato", 0.75),
    Product("Lettuce", 1.50), Product("Cucumber", 0.80), Product("Avocado", 2.00),
    Product("Mango", 1.75), Product("Pineapple", 3.50), Product("Watermelon", 5.00)
]

# Generate synthetic sales data
corp.generate_sales_data(products)

# Convert sales report to a Pandas DataFrame and display
sales_df = corp.generate_sales_report()
print(sales_df.head())
```

```
                   Timestamp        Date      Time  StoreID  CustomerID  \
0 2024-11-23 12:05:19.146058  2024-11-23  12:05:19        1        1001
1 2024-11-21 12:57:19.146088  2024-11-21  12:57:19        1        1002
2 2024-11-20 21:17:19.146101  2024-11-20  21:17:19        1        1003
3 2024-11-20 11:19:19.146112  2024-11-20  11:19:19        1        1004
4 2024-11-21 09:35:19.146127  2024-11-21  09:35:19        1        1005

   OrderID                                          Products  TotalPrice
0     9945  [Potato, Orange, Avocado, Broccoli, Grapes, Pi...       23.50
1     7450  [Tomato, Pineapple, Cucumber, Strawberry, Banana]        9.30
2     4237               [Broccoli, Cucumber, Orange, Apple]        5.05
3     9848                                 [Avocado, Banana]        2.75
4     2817  [Avocado, Grapes, Orange, Apple, Carrot, Brocc...       17.25
```

Double-click (or enter) to edit

```
import pandas as pd
from collections import defaultdict

# Goal: Analyze sales data to identify best-selling items per store and across the entire corporation using market basket analysis principle

# ... (Existing code from previous responses) ...

def analyze_best_selling_items(sales_df):
    """Analyzes sales data to find best-selling items."""

    # Analyze best-selling items per store
    store_bestsellers = {}
    for store_id in sales_df["StoreID"].unique():
        store_data = sales_df[sales_df["StoreID"] == store_id]
        product_counts = defaultdict(int)
        for _, row in store_data.iterrows():
            for product in row["Products"]:
                product_counts[product] += 1
        store_bestsellers[store_id] = dict(sorted(product_counts.items(), key=lambda item: item[1], reverse=True))

    # Analyze best-selling items across the entire corporation
```

```
        overall_product_counts = defaultdict(int)
        for _, row in sales_df.iterrows():
          for product in row["Products"]:
              overall_product_counts[product] += 1
        overall_bestsellers = dict(sorted(overall_product_counts.items(), key=lambda item: item[1], reverse=True))

        return store_bestsellers, overall_bestsellers


    # Analyze sales data
    store_bestsellers, overall_bestsellers = analyze_best_selling_items(sales_df)

    # Print results
    print("\nBest-selling items per store:")
    for store_id, products in store_bestsellers.items():
        print(f"\nStore {store_id}:")
        for product, count in products.items():
            print(f"  {product}: {count}")

    print("\nOverall best-selling items:")
    for product, count in overall_bestsellers.items():
        print(f"  {product}: {count}")
```

```
Best-selling items per store:

Store 1:
  Avocado: 10
  Potato: 8
  Orange: 8
  Broccoli: 7
  Pineapple: 7
  Strawberry: 7
  Watermelon: 7
  Apple: 7
  Cucumber: 7
  Mango: 7
  Banana: 6
  Carrot: 6
  Tomato: 5
  Grapes: 4
  Lettuce: 4

Store 2:
  Mango: 10
  Tomato: 10
  Watermelon: 10
  Grapes: 10
  Potato: 9
  Cucumber: 9
  Avocado: 9
  Pineapple: 9
  Apple: 9
  Broccoli: 9
  Orange: 8
  Strawberry: 8
  Lettuce: 8
  Carrot: 7
  Banana: 6

Store 3:
  Orange: 10
  Apple: 9
  Lettuce: 9
  Watermelon: 9
  Avocado: 9
  Cucumber: 9
  Banana: 9
  Broccoli: 8
  Mango: 8
  Carrot: 8
  Potato: 8
  Tomato: 7
  Pineapple: 7
  Grapes: 7
  Strawberry: 7

Store 4:
  Mango: 10
  Cucumber: 9
  Pineapple: 9
```