Project 1

```
import random
import csv
from datetime import datetime

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

    def __str__(self):
        return f"{self.name}: ${self.price:.2f}"



class Order:
    def __init__(self, order_id):
        self.order_id = order_id
        self.products = []

    def add_product(self, product):
        self.products.append(product)

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

    def __str__(self):
        product_list = ', '.join([str(product) for product in self.products])
        return f"Order ID: {self.order_id}, Products: [{product_list}], Total: ${self.get_total_price():.2f}"



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

    def create_order(self):
        order_id = random.randint(1000, 9999)  # Random Order ID
        order = Order(order_id)
        self.orders.append(order)
        return order

    def __str__(self):
        return f"Customer ID: {self.customer_id}"



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

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

    def __str__(self):
        return f"Store ID: {self.store_id}"



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

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



# Sample product list to choose from
sample_products = [
    Product("Laptop", 999.99),
    Product("Smartphone", 499.99),
    Product("Headphones", 79.99),
    Product("Mouse", 25.99),
    Product("Keyboard", 45.99),
    Product("Monitor", 199.99)
]

# Initialize the Corporation
corp = Corporation()

# Create stores
for store_id in range(1, 6):  # Creating 5 stores for example
    store = Store(store_id)
    corp.add_store(store)

    # Simulate customer visits
    for customer_id in range(1, 21):  # Each store has 20 customers visiting in a year
        customer = Customer(customer_id)
        store.add_customer(customer)

        # Simulate multiple orders per customer
        for visit in range(random.randint(1, 5)):  # Each customer visits 1 to 5 times
            order = customer.create_order()

            # Add random products to the order
            for _ in range(random.randint(1, 5)):  # Each order contains 1 to 5 products
                order.add_product(random.choice(sample_products))

# Write sales data to a CSV file
with open('sales_data.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    # Writing the header
    writer.writerow(['Date', 'Time', 'StoreID', 'CustomerID', 'OrderID', 'Product Name', 'Price'])

    # Writing data for each sale
    for store in corp.stores:
        for customer in store.customers:
            for order in customer.orders:
                date_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S").split(' ')
                date = date_time[0]
                time = date_time[1]
                for product in order.products:
                    writer.writerow([date, time, store.store_id, customer.customer_id, order.order_id, product.name, product.price])



from google.colab import files
files.download('sales_data.csv')
```

Project 2 Visualizations

```
    from google.colab import files
    uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable

```
    import pandas as pd

    # Replace 'your_file.csv' with the name of your file
    df = pd.read_csv('sales_data.csv')



    # Group by OrderID and get the list of products in each order
    order_products = df.groupby('OrderID')['Product Name'].apply(list)

    # Count the frequency of each product across all orders
    product_frequency = pd.Series([product for sublist in order_products for product in sublist]).value_counts()

    # Display the most prevalent products
    print("Most prevalent products:")
    print(product_frequency.head(10))  # Top 10 products
```

```
Most prevalent products:
Monitor       188
Headphones    170
Keyboard      152
Smartphone    150
Laptop        150
Mouse         135
Name: count, dtype: int64
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
```

```
# Define a threshold for a "large basket" (e.g., more than 3 items)
large_basket_threshold = 3

# Calculate basket sizes for each order
basket_sizes = df.groupby('OrderID')['Product Name'].count()

# Identify large baskets
large_baskets = basket_sizes[basket_sizes > large_basket_threshold]

# Calculate the frequency of large baskets
large_basket_frequency = len(large_baskets) / len(basket_sizes)

print(f"Frequency of large baskets (more than {large_basket_threshold} items): {large_basket_frequency:.2f}")

# Identify large buyers (customers with multiple large baskets)
large_buyers = df[df['OrderID'].isin(large_baskets.index)].groupby('CustomerID')['OrderID'].nunique()
large_buyers = large_buyers[large_buyers > 1]  # Customers with more than 1 large basket

print("Large buyers (customers with multiple large baskets):")
print(large_buyers)
```

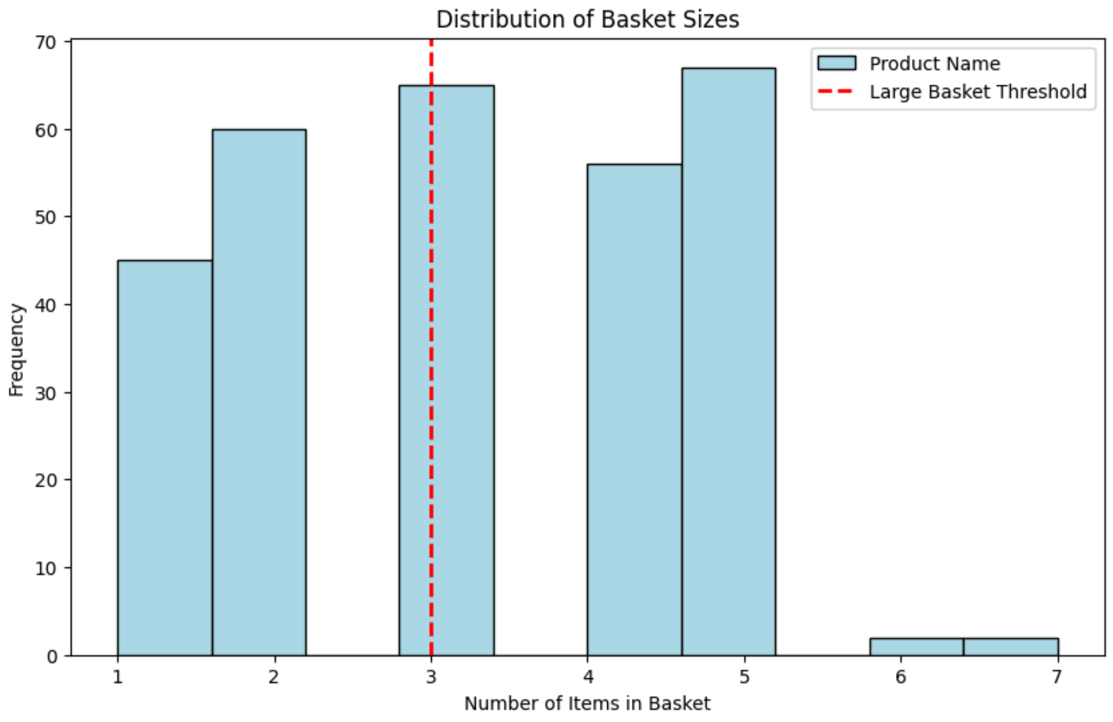
```
Frequency of large baskets (more than 3 items): 0.43
Large buyers (customers with multiple large baskets):
CustomerID
1      4
2      3
3      8
4      4
5      6
6      3
7      8
8      9
9      9
10     7
11     9
12     8
13    12
15     6
16     5
17     8
18     6
19     6
20    11
Name: OrderID, dtype: int64
```

```
# Filter the data for large basket orders
large_basket_orders = df[df['OrderID'].isin(large_baskets.index)]

# Group by StoreID and count the number of large baskets
store_large_baskets = large_basket_orders.groupby('StoreID')['OrderID'].nunique()

print("Stores containing large-basket buyers and their frequency:")
print(store_large_baskets)
```

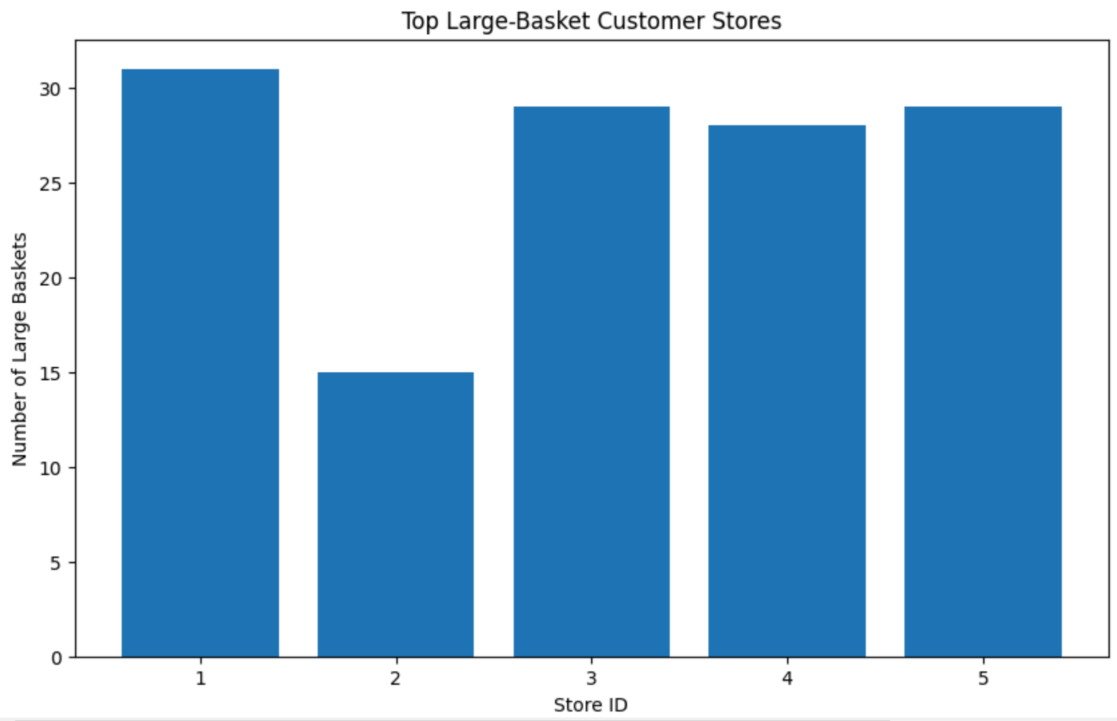
```
Stores containing large-basket buyers and their frequency:
StoreID
1    31
2    15
3    29
4    28
5    29
Name: OrderID, dtype: int64
```

```
# 1. Histogram of Basket Sizes
plt.figure(figsize=(10, 6))
basket_sizes.plot(kind='hist', bins=10, color='lightblue', edgecolor='black')
plt.title('Distribution of Basket Sizes')
plt.xlabel('Number of Items in Basket')
plt.ylabel('Frequency')
plt.axvline(large_basket_threshold, color='red', linestyle='dashed', linewidth=2, label='Large Basket Threshold')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
```



```
import matplotlib.pyplot as plt

# Sort the stores by large basket frequency
store_large_baskets = store_large_baskets.sort_values(ascending=False)

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(store_large_baskets.index, store_large_baskets.values)
plt.xlabel("Store ID")
plt.ylabel("Number of Large Baskets")
plt.title("Top Large-Basket Customer Stores")
plt.xticks(store_large_baskets.index)
plt.show()
```

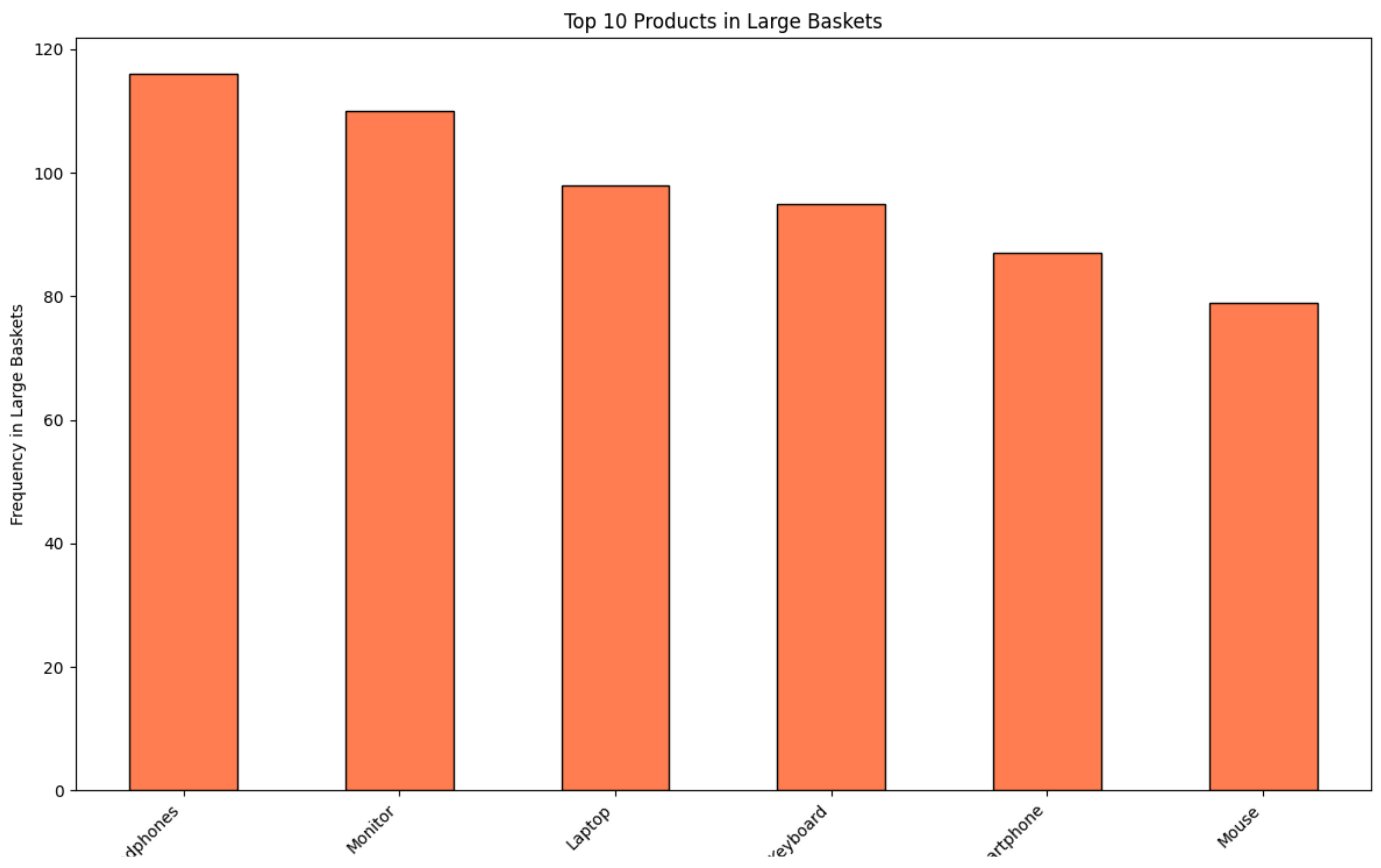


```
# Get the products in large baskets
large_basket_products = large_basket_orders.groupby('OrderID')['Product Name'].apply(list)

# Count the frequency of each product in large baskets
product_frequency_large_baskets = pd.Series([product for sublist in large_basket_products for product in sublist]).value_counts()

print("Top products in large baskets:")
print(product_frequency_large_baskets.head(10))  # Top 10 products
```

```
Top products in large baskets:
Headphones    116
Monitor       110
Laptop         98
Keyboard       95
Smartphone     87
Mouse          79
Name: count, dtype: int64
```

```
import matplotlib.pyplot as plt

# Plot the top 10 products in large baskets
plt.figure(figsize=(12, 8))
product_frequency_large_baskets.head(10).plot(kind='bar', color='coral', edgecolor='black')

# Add titles and labels
plt.title('Top 10 Products in Large Baskets')
plt.xlabel('Product Name')
plt.ylabel('Frequency in Large Baskets')
plt.xticks(rotation=45, ha='right')

# Display the plot
plt.tight_layout()  # Adjust layout to prevent label cutoff
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell`
  and should_run_async(code)
```



```
# Assuming '
```

Project 3

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

# Load the sales data
df = pd.read_csv('sales_data.csv')

# 1. Best-selling items for each store:
def get_best_selling_items_per_store(df, store_id):
  """
  Returns the best-selling items for a specific store.
  """
  store_data = df[df['StoreID'] == store_id]
  best_selling_items = store_data['Product Name'].value_counts().head(5)  # Top 5 items
  return best_selling_items

# Get best-selling items for each store
for store_id in df['StoreID'].unique():
  print(f"Best-selling items for Store {store_id}:")
  print(get_best_selling_items_per_store(df, store_id))
  print("-" * 20)

# 2. Best-selling items across the entire organization:
best_selling_items_overall = df['Product Name'].value_counts().head(10)  # Top 10 items
print("Best-selling items across the entire organization:")
print(best_selling_items_overall)
print("-" * 20)

# 3. Market Basket Analysis:
# Create a one-hot encoded DataFrame for market basket analysis
basket = df.groupby(['OrderID', 'Product Name'])['Price'].sum().unstack().reset_index().fillna(0)
basket = pd.get_dummies(basket, columns=basket.columns[1:], prefix='', prefix_sep='').astype(bool)

# Apply Apriori algorithm to find frequent itemsets
frequent_itemsets = apriori(basket, min_support=0.05, use_colnames=True)
```

```
# Generate association rules - passing num_itemsets
num_itemsets = len(frequent_itemsets)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1, num_itemsets=num_itemsets

# Display the top 10 rules
print("Top 10 Association Rules:")
print(rules.head(10))
```

```
Best-selling items for Store 1:
Product Name
Monitor       54
Headphones    35
Laptop        32
Mouse         28
Keyboard      28
Name: count, dtype: int64
--------------------
Best-selling items for Store 2:
Product Name
Monitor       29
Smartphone    25
Headphones    23
Laptop        22
Keyboard      19
Name: count, dtype: int64
--------------------
Best-selling items for Store 3:
Product Name
Keyboard      45
Monitor       45
Headphones    36
Smartphone    34
Mouse         31
Name: count, dtype: int64
--------------------
Best-selling items for Store 4:
Product Name
Keyboard      35
Monitor       34
Smartphone    32
Laptop        32
Headphones    29
Name: count, dtype: int64
--------------------
Best-selling items for Store 5:
Product Name
Headphones    47
Laptop        37
Mouse         36
Smartphone    33
Monitor       26
Name: count, dtype: int64
--------------------
Best-selling items across the entire organization:
Product Name
Monitor       188
Headphones    170
Keyboard      152
Laptop        150
Smartphone    150
Mouse         135
Name: count, dtype: int64
--------------------
Top 10 Association Rules:
```