Project 2 Section

```
import pandas as pd

df = pd.read_csv('sales_data.csv')
```

```
/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)
```

```
print(df.head(10))
```

```
         Date      Time    StoreID CustomerID  \
0  2023-01-01  00:00:00  Store_001   Cust_842
1  2023-01-01  00:00:00  Store_001   Cust_271
2  2023-01-01  00:00:00  Store_001   Cust_271
3  2023-01-01  00:00:00  Store_001   Cust_271
4  2023-01-01  00:00:00  Store_001   Cust_768
5  2023-01-01  00:00:00  Store_001   Cust_154
6  2023-01-01  00:00:00  Store_001   Cust_832
7  2023-01-01  00:00:00  Store_001   Cust_832
8  2023-01-01  00:00:00  Store_001   Cust_832
9  2023-01-01  00:00:00  Store_001   Cust_857

                                OrderID Product Name   Price
0  bc2cbca2-cf74-49de-b096-af3c6a7575f5      Monitor   300.0
1  8f7ac0dc-563e-4495-b7d1-8728c5460716      Printer   150.0
2  8f7ac0dc-563e-4495-b7d1-8728c5460716        Mouse    25.0
3  8f7ac0dc-563e-4495-b7d1-8728c5460716      Monitor   300.0
4  8f9d7a8d-85c8-48d9-99a7-4d1e3ef57f71     Keyboard    45.0
5  1af72663-788b-4ded-a3ef-e8df00c65b55      Printer   150.0
6  38437756-7583-4f2f-82f4-dcf6c52bc62e       Laptop  1200.0
7  38437756-7583-4f2f-82f4-dcf6c52bc62e      Monitor   300.0
8  38437756-7583-4f2f-82f4-dcf6c52bc62e      Printer   150.0
9  a48a7304-fe97-45e2-bf2d-5c40f96f824d      Printer   150.0
/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 pandas as pd
import matplotlib.pyplot as plt
```

```
/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)
```

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

```
/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)
```

```
# Display the first few rows of the dataframe
print(sales_data.head())
```

```
         Date      Time    StoreID CustomerID  \
0  2023-01-01  00:00:00  Store_001   Cust_842
1  2023-01-01  00:00:00  Store_001   Cust_271
2  2023-01-01  00:00:00  Store_001   Cust_271
3  2023-01-01  00:00:00  Store_001   Cust_271
4  2023-01-01  00:00:00  Store_001   Cust_768

                                OrderID Product Name  Price
0  bc2cbca2-cf74-49de-b096-af3c6a7575f5      Monitor  300.0
1  8f7ac0dc-563e-4495-b7d1-8728c5460716      Printer  150.0
2  8f7ac0dc-563e-4495-b7d1-8728c5460716        Mouse   25.0
3  8f7ac0dc-563e-4495-b7d1-8728c5460716      Monitor  300.0
4  8f9d7a8d-85c8-48d9-99a7-4d1e3ef57f71     Keyboard   45.0
/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)
```

```
# 1. The most prevalent products in customer baskets
prevalent_products = sales_data['Product Name'].value_counts()
```

```
print("Most prevalent products in customer baskets:")
print(prevalent_products)
```

```
Most prevalent products in customer baskets:
Product Name
Laptop      226
Mouse       208
Keyboard    195
Monitor     188
Printer     182
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)
```

```
# 2. The frequency by which customers were large buyers or filled up large baskets
# Assuming large basket is defined as having more than 3 items in a single order
large_basket_orders = sales_data.groupby('OrderID').size()
large_basket_frequency = large_basket_orders[large_basket_orders > 3].count()
print(f"Frequency of large basket orders: {large_basket_frequency}")
```

```
Frequency of large basket orders: 0
/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)
```

```
# 3. Which stores contained the large-basket buyers and by how much
large_basket_store_counts = sales_data[sales_data['OrderID'].isin(large_basket_orders[large_basket_orders > 3].index)]['StoreID'].value_coun
print("Stores containing large-basket buyers and their counts:")
print(large_basket_store_counts)
```

```
Stores containing large-basket buyers and their counts:
Series([], 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)
```

```
# 4. A visualization that ranks the top large-basket customer stores by frequency
plt.figure(figsize=(10, 6))
if not large_basket_store_counts.empty:
    large_basket_store_counts.plot(kind='bar')
    plt.title('Top Large-Basket Customer Stores by Frequency')
    plt.xlabel('Store ID')
    plt.ylabel('Frequency')
    plt.xticks(rotation=45)
    plt.show()
else:
    print("No large-basket orders found to plot.")
```

```
No large-basket orders found to plot.
/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)
<Figure size 1000x600 with 0 Axes>
```

```
# 5. A top-n list of products which were typical to customers in this demographic
top_n_products = prevalent_products.head(10)
print("Top-N list of products typical to customers in this demographic:")
print(top_n_products)
```

```
/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)
Top-N list of products typical to customers in this demographic:
Product Name
Laptop      226
Mouse       208
Keyboard    195
Monitor     188
Printer     182
Name: count, dtype: int64
```

```
# 6. A categorical approach to the above demographic – what is the categoric makeup of their baskets on average?
categoric_makeup = sales_data[sales_data['OrderID'].isin(large_basket_orders[large_basket_orders > 3].index)]['Product Name'].value_counts(n
print("Categoric makeup of their baskets on average:")
print(categoric_makeup)
```

```
/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)
Categoric makeup of their baskets on average:
Series([], Name: proportion, dtype: float64)
```

```
# 7. Formulate a visualization for item 6
plt.figure(figsize=(10, 6))
if not categoric_makeup.empty:
    categoric_makeup.plot(kind='bar')
    plt.title('Categoric Makeup of Large-Basket Orders')
    plt.xlabel('Product Name')
    plt.ylabel('Proportion')
    plt.xticks(rotation=45)
    plt.show()
else:
    print("No large-basket orders found to plot categoric makeup.")
```

```
/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)
No large-basket orders found to plot categoric makeup.
<Figure size 1000x600 with 0 Axes>
```

Project 3 Section

1. Most Prevalent Products

```
# Count the occurrences of each product (PROJECT 3)
product_counts = sales_data['Product Name'].value_counts()

# Display the most prevalent products
print("Most Prevalent Products:")
print(product_counts.head())
```

```
Most Prevalent Products:
Product Name
Laptop      226
Mouse       208
Keyboard    195
Monitor     188
Printer     182
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)
```

2. Large Basket Orders

```
# Define a large basket as having more than a certain number of items (PROJECT 3)
large_basket_threshold = 5

# Group by CustomerID and count the number of items in each basket
basket_sizes = sales_data.groupby('CustomerID').size()

# Count the number of large baskets
large_basket_counts = basket_sizes[basket_sizes > large_basket_threshold].count()

# Display the frequency of large buyers
print("Number of Large Baskets:")
print(large_basket_counts)
```

```
/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)
Number of Large Baskets:
18
```

3. Stores Containing Large-Basket Buyers

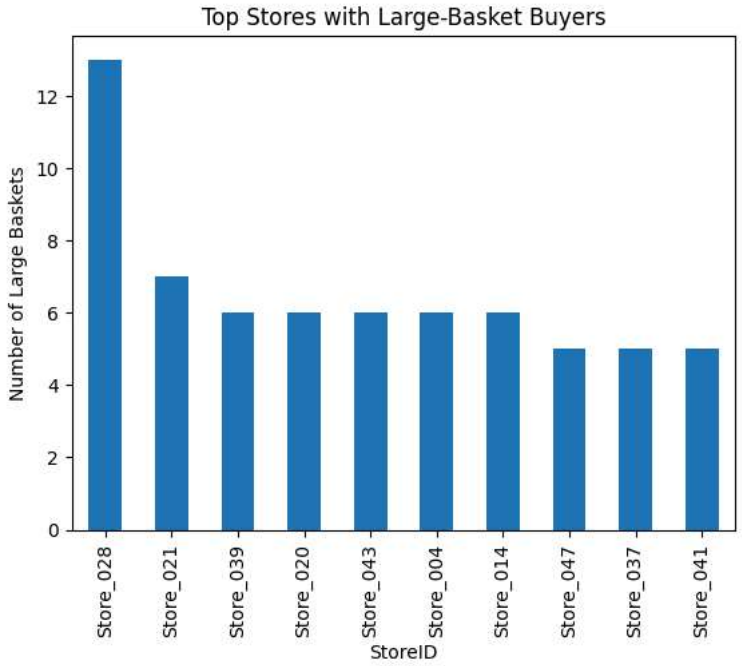
```
# Group by StoreID and count the number of large baskets in each store (PROJECT 3)
large_basket_stores = sales_data[sales_data['CustomerID'].isin(basket_sizes[basket_sizes > large_basket_threshold].index)]
store_large_basket_counts = large_basket_stores['StoreID'].value_counts()

# Display the stores with large-basket buyers
print("Stores with Large-Basket Buyers:")
print(store_large_basket_counts)
```

```
/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)
Stores with Large-Basket Buyers:
StoreID
Store_028    13
Store_021     7
Store_039     6
Store_020     6
Store_043     6
Store_004     6
Store_014     6
Store_047     5
Store_037     5
Store_041     5
Store_029     4
Store_017     4
Store_011     4
Store_008     4
Store_038     3
Store_034     3
Store_035     3
Store_003     3
Store_032     3
Store_030     3
Store_026     3
Store_015     3
Store_010     3
Store_009     3
Store_044     2
Store_045     2
Store_046     1
Store_007     1
Store_051     1
Name: count, dtype: int64
```

4. Visualization of Top Large-Basket Customer Stores by Frequency

```
import matplotlib.pyplot as plt

# Plot the top stores with large-basket buyers (PROJECT 3)
store_large_basket_counts.head(10).plot(kind='bar')
plt.title('Top Stores with Large-Basket Buyers')
plt.xlabel('StoreID')
plt.ylabel('Number of Large Baskets')
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)
```



5. Top-N List of Products Typical to Customers

```
# Get the products in large baskets (PROJECT 3)
large_basket_products = large_basket_stores['Product Name'].value_counts()

# Display the top-N products
top_n = 10
print("Top Products in Large Baskets:")
print(large_basket_products.head(top_n))
```

```
Top Products in Large Baskets:
Product Name
Keyboard    27
Printer     27
Laptop      22
Monitor     21
Mouse       21
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)
```

6. Categoric Makeup of Their Baskets on Average

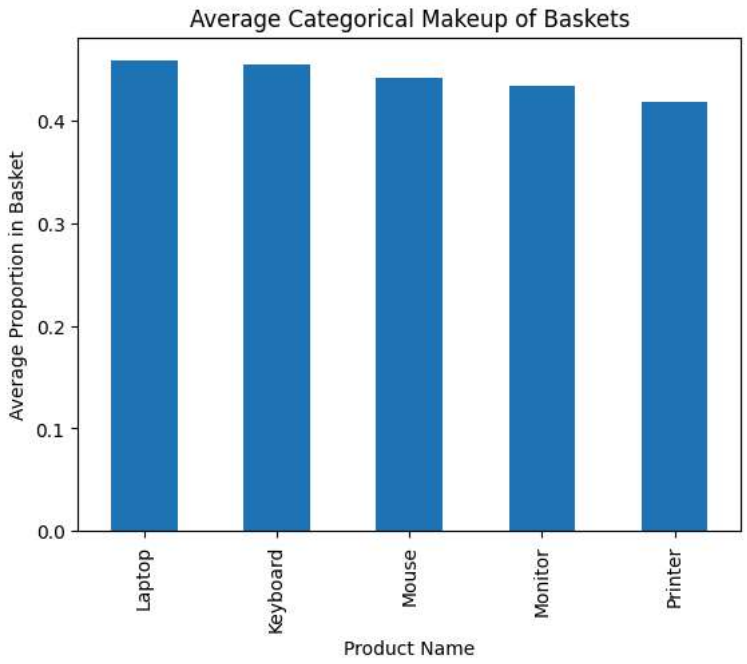
```
# Group by CustomerID and get the average basket makeup
basket_makeup = sales_data.groupby('CustomerID')['Product Name'].apply(lambda x: x.value_counts(normalize=True))

# Display the average categorical makeup of baskets
print("Average Categorical Makeup of Baskets:")
print(basket_makeup.head())
```

```
/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)
Average Categorical Makeup of Baskets:
CustomerID
Cust_003    Keyboard    1.000000
Cust_004    Laptop      1.000000
Cust_005    Laptop      0.333333
            Printer     0.333333
            Mouse       0.333333
Name: Product Name, dtype: float64
```

7. Visualization for Categoric Makeup

```
# Plot the categorical makeup of baskets
basket_makeup_df = basket_makeup.unstack().mean().sort_values(ascending=False)
basket_makeup_df.plot(kind='bar')
plt.title('Average Categorical Makeup of Baskets')
plt.xlabel('Product Name')
plt.ylabel('Average Proportion in Basket')
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)
```



8. Market Basket Analysis

```
from mlxtend.frequent_patterns import apriori, association_rules

# Create a basket for each store
basket = (sales_data.groupby(['StoreID', 'OrderID', 'Product Name'])['Product Name']
          .count().unstack().reset_index().fillna(0)
          .set_index(['StoreID', 'OrderID']))

# Convert the values to 1 and 0
def encode_units(x):
    return 1 if x >= 1 else 0

basket_sets = basket.applymap(encode_units)

# Perform market basket analysis using the Apriori algorithm
frequent_itemsets = apriori(basket_sets, min_support=0.01, use_colnames=True)

# Generate the association rules, specifying num_itemsets
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1, support_only=False, num_itemsets=frequent_itemsets['itemsets'].a
# Display the most frequently occurring itemsets
print(frequent_itemsets.sort_values(by='support', ascending=False).head())

# Display the association rules
print(rules.head())
```

```
     support    itemsets
1  0.446640    (Laptop)
3  0.411067     (Mouse)
0  0.385375  (Keyboard)
```

```
2  0.371542   (Monitor)
4  0.359684   (Printer)
Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent support, support, confidence, lift, representativity, leverage, convi
Index: []
/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)
<ipython-input-62-280dec10de1d>:12: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  basket_sets = basket.applymap(encode_units)
/usr/local/lib/python3.10/dist-packages/mlxtend/frequent_patterns/fpcommon.py:161: DeprecationWarning: DataFrames with non-bool types re
  warnings.warn(
```