In [52]:

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

In [53]:

```
# Load the Data
data = pd.read_csv('sales_data.csv')
data.head()
```

Out[53]:

	Date	Time	StoreID	CustomerID	OrderID	Product Name	Price
0	2024-11-21	14:44:53	1	1	3084	Product C	7.99
1	2024-11-21	14:44:53	1	1	3084	Product H	15.99
2	2024-11-21	14:44:53	1	1	3084	Product I	4.99
3	2024-11-21	14:44:53	1	1	3084	Product B	5.49
4	2024-11-21	14:44:53	1	1	3084	Product F	8.99

In [54]:

```
# Create a DataFrame
df = pd.DataFrame(data, columns=['Date', 'Time', 'StoreID', 'CustomerID', 'OrderID'
```

In [55]:

```
df.info
```

Out[55]:

```
<bound method DataFrame.info of                 Date      Time  StoreID  CustomerID
OrderID Product Name  Price
0        2024-11-21  14:44:53        1           1     3084  Product C   7.99
1        2024-11-21  14:44:53        1           1     3084  Product H  15.99
2        2024-11-21  14:44:53        1           1     3084  Product I   4.99
3        2024-11-21  14:44:53        1           1     3084  Product B   5.49
4        2024-11-21  14:44:53        1           1     3084  Product F   8.99
...             ...       ...      ...         ...      ...        ...    ...
600002   2024-11-21  14:44:56      100         100     3419  Product B   5.49
600003   2024-11-21  14:44:56      100         100     3419  Product G   6.49
600004   2024-11-21  14:44:56      100         100     3419  Product C   7.99
600005   2024-11-21  14:44:56      100         100     3242  Product C   7.99
600006   2024-11-21  14:44:56      100         100     3242  Product C   7.99

[600007 rows x 7 columns]>
```

In [56]:

```
df.columns
```

Out[56]:

```
Index(['Date', 'Time', 'StoreID', 'CustomerID', 'OrderID', 'Product Name',
       'Price'],
      dtype='object')
```

In [57]:

```
store_count = df ['StoreID'].nunique()
print(f"Number of Unique stores: {store_count}")
```

```
Number of Unique stores: 100
```

In [58]:

```
# Most prevelant products in baskets
prevalent_products = df['Product Name'].value_counts().head(5)
print(f"Top 5 of our Most Prevalent Products: {prevalent_products}")
```

```
Top 5 of our Most Prevalent Products: Product Name
Product I    60363
Product H    60316
Product C    60079
Product J    60079
Product B    60029
Name: count, dtype: int64
```

In [59]:

```python
# Did not have qauntity so I Grouped by StoreID, CustomerID, Date, and Time to find
grouped_data = data.groupby(['StoreID', 'CustomerID', 'Date', 'Time']).size().reset

# Display the first few rows of the grouped data
grouped_data.head()
```

Out[59]:

	StoreID	CustomerID	Date	Time	ItemCount
0	1	1	2024-11-21	14:44:53	69
1	1	2	2024-11-21	14:44:53	52
2	1	3	2024-11-21	14:44:53	69
3	1	4	2024-11-21	14:44:53	65
4	1	5	2024-11-21	14:44:53	66

In [60]:

```python
# Define How much a Large Basket is (7)
large_basket_threshold = 7

# Classify baskets as large or small
grouped_data['BasketType'] = grouped_data['ItemCount'].apply(lambda x: 'Large' if x

# Assign a unique BasketID to each transaction
grouped_data['BasketID'] = range(1, len(grouped_data) + 1)

# Filter transactions for large baskets
large_basket_transactions = grouped_data[grouped_data['BasketType'] == 'Large']

# Filter transactions for small baskets
small_basket_transactions = grouped_data[grouped_data['BasketType'] == 'Small']

# Merge original data with classified basket data
data_with_baskets = data.merge(grouped_data[['StoreID', 'CustomerID', 'Date', 'Time
print(data_with_baskets.head())
```

```
         Date      Time  StoreID  CustomerID  OrderID Product Name  Price  \
0  2024-11-21  14:44:53        1           1     3084    Product C   7.99
1  2024-11-21  14:44:53        1           1     3084    Product H  15.99
2  2024-11-21  14:44:53        1           1     3084    Product I   4.99
3  2024-11-21  14:44:53        1           1     3084    Product B   5.49
4  2024-11-21  14:44:53        1           1     3084    Product F   8.99

  BasketType  BasketID
0      Large         1
1      Large         1
2      Large         1
3      Large         1
4      Large         1
```

In [61]:

```
# Filter transactions for large baskets
large_basket_transactions = grouped_data[grouped_data['ItemCount'] >= large_basket_

# All of large-basket transactions
print(f"Transactions with large baskets (7 or more items):\n {large_basket_transact
```

```
Transactions with large baskets (7 or more items):
        StoreID  CustomerID        Date      Time  ItemCount BasketType  \
0            1           1  2024-11-21  14:44:53         69      Large
1            1           2  2024-11-21  14:44:53         52      Large
2            1           3  2024-11-21  14:44:53         69      Large
3            1           4  2024-11-21  14:44:53         65      Large
4            1           5  2024-11-21  14:44:53         66      Large
...        ...         ...         ...       ...        ...        ...
9998       100          96  2024-11-21  14:44:56         61      Large
9999       100          97  2024-11-21  14:44:56         53      Large
10000      100          98  2024-11-21  14:44:56         61      Large
10001      100          99  2024-11-21  14:44:56         58      Large
10002      100         100  2024-11-21  14:44:56         50      Large

       BasketID
0             1
1             2
2             3
3             4
4             5
...         ...
9998       9999
9999      10000
10000     10001
10001     10002
10002     10003

[10003 rows x 7 columns]
```

In [62]:

```
# Calculate the frequency of large-basket transactions
large_basket_frequency = large_basket_transactions.shape[0]
print(f"Frequency of large-basket transactions (7 or more items): {large_basket_fre

# Group by StoreID and count the number of large-basket transactions for each store
large_basket_stores = large_basket_transactions.groupby('StoreID').size().reset_ind
```

```
Frequency of large-basket transactions (7 or more items): 10003
```

In [63]:

```
# Sort stores showing large-basket buyers in descending order
large_basket_stores_sorted = large_basket_stores.sort_values(by='LargeBasketCount',

# Display the top 10 stores with large-basket buyers
top_10_large_basket_stores = large_basket_stores_sorted.head(10)
print(f"Top 10 stores with large-basket buyers and their counts:\n{top_10_large_bas
```

```
Top 10 stores with large-basket buyers and their counts:
    StoreID  LargeBasketCount
29       30               101
59       60               101
89       90               101
0         1               100
3         4               100
4         5               100
6         7               100
5         6               100
8         9               100
9        10               100
```

In [64]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

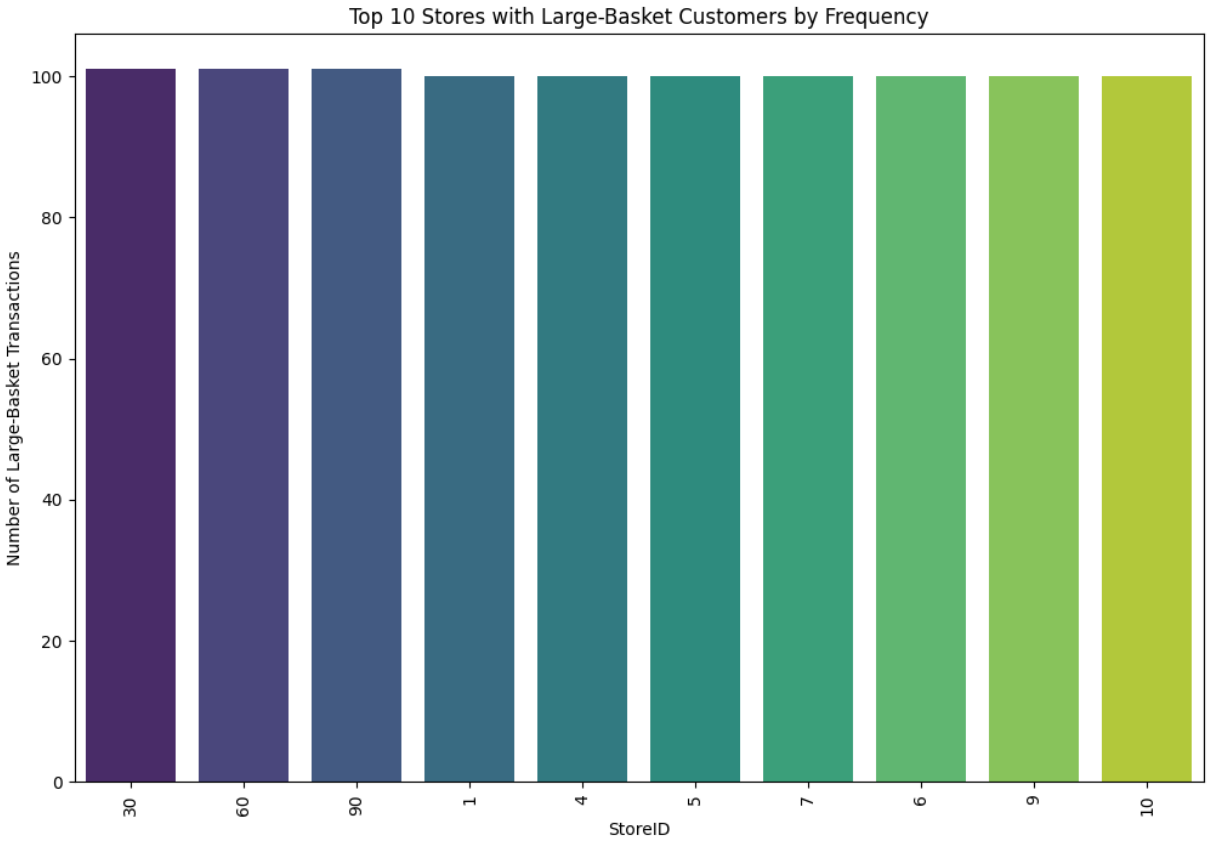
In [65]:

```
# Visualization of Top 10 Stores By Large-Basket Frequency
plt.figure(figsize=(12, 8))
sns.barplot(x='StoreID', y='LargeBasketCount', data=top_10_large_basket_stores, pal
plt.xlabel('StoreID')
plt.ylabel('Number of Large-Basket Transactions')
plt.title('Top 10 Stores with Large-Basket Customers by Frequency')
plt.xticks(rotation=90)
plt.show()
```

```
C:\Users\eriks\AppData\Local\Temp\ipykernel_5448\1057542087.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14
.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='StoreID', y='LargeBasketCount', data=top_10_large_basket_stores, pa
lette='viridis', order=top_10_large_basket_stores['StoreID'])
```



In [66]:

```
# Filter original data for large-basket transactions
large_basket_data = data.merge(large_basket_transactions, on=['StoreID', 'CustomerI

# Group by Product Name and count the number of occurrences
large_basket_product_frequency = large_basket_data['Product Name'].value_counts().r
large_basket_product_frequency.columns = ['Product Name', 'Frequency']

# Number of Products shown
top_n = 10

# Get the top-n products
top_n_large_basket_products = large_basket_product_frequency.head(top_n)

# Display the top-n products for large-basket transactions
print(f"Top {top_n} products for large-basket customers:\n{top_n_large_basket_produ
```

```
Top 10 products for large-basket customers:
  Product Name  Frequency
0    Product I      60363
1    Product H      60316
2    Product C      60079
3    Product J      60079
4    Product B      60029
5    Product D      60017
6    Product A      59826
7    Product G      59805
8    Product F      59779
9    Product E      59714
```

In [67]:

```
# Created categories for each of my products
```

```
product_to_category = {
    'Product A': 'Category1',
    'Product B': 'Category2',
    'Product C': 'Category3',
    'Product D': 'Category4',
    'Product E': 'Category5',
    'Product F': 'Category6',
    'Product G': 'Category7',
    'Product H': 'Category8',
    'Product I': 'Category9',
    'Product J': 'Category10',

}

# Add a 'Category' column to your data
data['Category'] = data['Product Name'].map(product_to_category)
print(data.head())
```

```
         Date      Time  StoreID  CustomerID  OrderID Product Name  Price  \
0  2024-11-21  14:44:53        1           1     3084    Product C   7.99
1  2024-11-21  14:44:53        1           1     3084    Product H  15.99
2  2024-11-21  14:44:53        1           1     3084    Product I   4.99
3  2024-11-21  14:44:53        1           1     3084    Product B   5.49
4  2024-11-21  14:44:53        1           1     3084    Product F   8.99

    Category
0  Category3
1  Category8
2  Category9
3  Category2
4  Category6
```

In [68]:

```
# Filter original data for large-basket transactions
large_basket_data = data.merge(large_basket_transactions, on=['StoreID', 'CustomerI
print(large_basket_data.head())


# Group by Category and calculate the average number of items per category
category_average = large_basket_data.groupby('Category').size().reset_index(name='A

# Display the categorical makeup of their baskets
print(f"Categorical makeup of their baskets on average:\n{category_average}")

# Create a bar chart for the categorical makeup of their baskets
plt.figure(figsize=(12, 8))
sns.barplot(x='Category', y='AverageItemCount', data=category_average, palette='vir
plt.xlabel('Category')
plt.ylabel('Average Number of Items')
plt.title('Categorical Makeup of Baskets on Average')
plt.xticks(rotation=90)
plt.show()
```

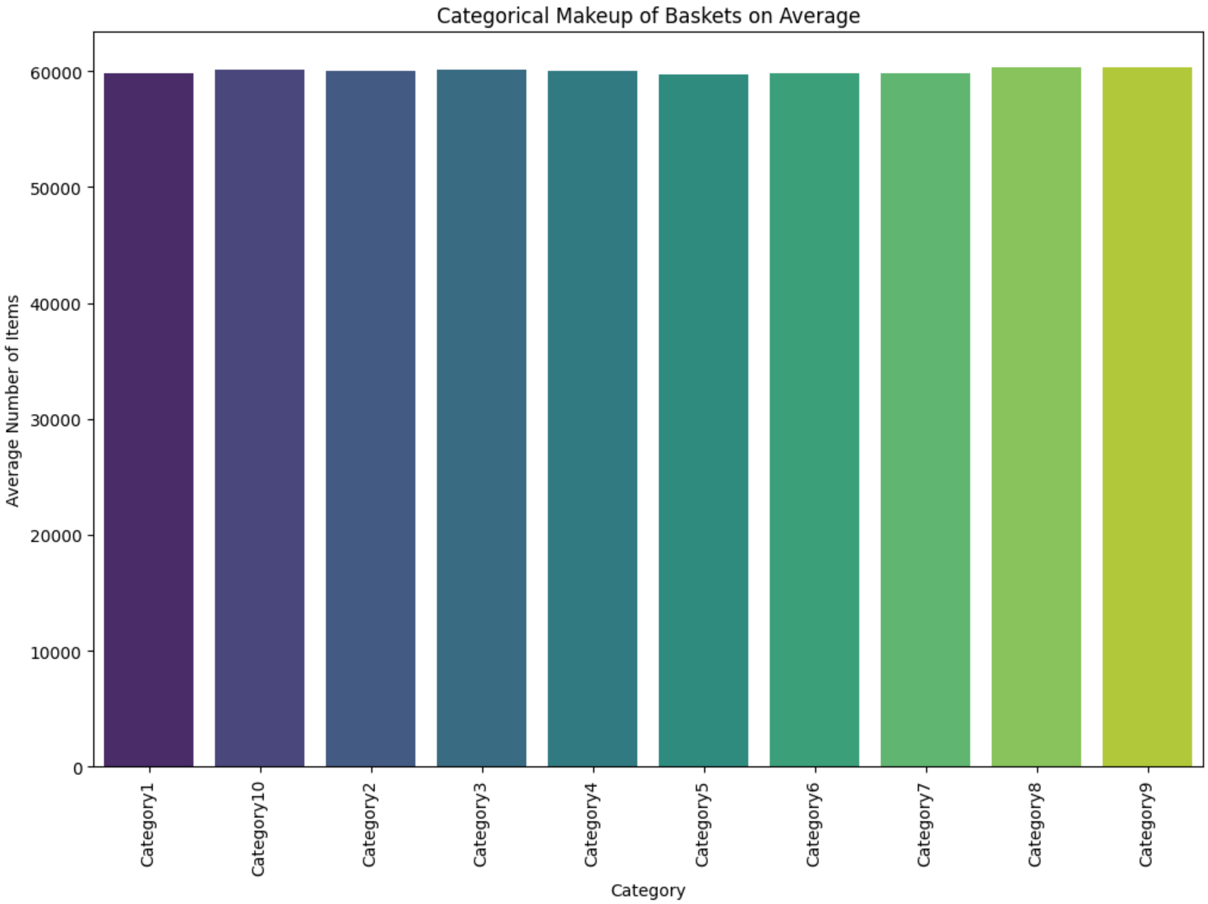
```
         Date      Time  StoreID  CustomerID  OrderID Product Name  Price  \
0  2024-11-21  14:44:53        1           1     3084    Product C   7.99   
1  2024-11-21  14:44:53        1           1     3084    Product H  15.99   
2  2024-11-21  14:44:53        1           1     3084    Product I   4.99   
3  2024-11-21  14:44:53        1           1     3084    Product B   5.49   
4  2024-11-21  14:44:53        1           1     3084    Product F   8.99   

    Category  ItemCount BasketType  BasketID  
0  Category3         69      Large         1  
1  Category8         69      Large         1  
2  Category9         69      Large         1  
3  Category2         69      Large         1  
4  Category6         69      Large         1  
Categorical makeup of their baskets on average:
     Category  AverageItemCount
0   Category1             59826
1  Category10             60079
2   Category2             60029
3   Category3             60079
4   Category4             60017
5   Category5             59714
6   Category6             59779
7   Category7             59805
8   Category8             60316
9   Category9             60363
```

```
C:\Users\eriks\AppData\Local\Temp\ipykernel_5448\923698545.py:14: FutureWarning: 

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14
.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='Category', y='AverageItemCount', data=category_average, palette='vi
ridis')
```


Categorical Makeup of Baskets on Average
Average Number of Items
60000
50000
40000
30000
20000
10000
0
Category1
Category10
Category2
Category3
Category4
Category5
Category6
Category7
Category8
Category9
Category