## Step 1: Create a New Notebook and Import Libraries and Data

In [26]:
```
import pandas as pd
import numpy as np

# Specify the correct file path
file_path = r'C:\Users\Asus\Music\Instacart Basket Analysis\Data\Prepared Data\orders_products_merge__updated.pkl'

# Load the ords_prods_merge dataframe from the pickle file
ords_prods_merge = pd.read_pickle(file_path)
```

## Step 2: Find the Aggregated Mean of the "order_number" Column Grouped by "department_id"

In [29]:
```
# Calculate the mean of "order_number" grouped by "department_id"
mean_order_number_by_dept = ords_prods_merge.groupby('department_id')['order_number'].mean().reset_index()

# Display the result
print(mean_order_number_by_dept)
```

```
    department_id  order_number
0               1     15.457687
1               2     17.277920
2               3     17.179756
3               4     17.811403
4               5     15.213779
5               6     16.439806
6               7     17.225773
7               8     15.340520
8               9     15.895474
9              10     20.197148
10             11     16.170828
11             12     15.887622
12             13     16.583304
13             14     16.757377
14             15     16.165037
15             16     17.663250
16             17     15.694469
17             18     19.310397
18             19     17.177343
19             20     16.473447
20             21     22.902379
```

## Step 3: Analyze the Result

## Analysis of Mean Order Number by Department

The table above gives us a glimpse into how often customers reorder items from different departments. Here's a breakdown of the key insights:

Department 21 tops the list with the highest average order number of 22.90. This suggests that customers frequently reorder products from this department, indicating either a strong preference for these products or a high necessity that keeps customers coming back.

Department 10 also shows a high level of repeat purchases, with an average order number of 20.20. This is another department where customers seem to return often, possibly due to the essential nature of the products offered.

Department 18 stands out with an average order number of 19.31, suggesting that it's also a popular choice for repeat purchases.

On the flip side, Department 5 (average order number 15.21) and Department 8 (average order number 15.34) have lower averages. This could mean that customers are less inclined to reorder from these departments, perhaps due to the type of products, less frequent need, or lower customer engagement.

These findings help us understand what customers prefer and how they shop. Departments with higher reorder rates might be good targets for special promotions or loyalty programs to further boost sales. Meanwhile, for those with lower reorder rates, it might be worth exploring ways to increase customer engagement or understanding why customers aren't coming back as frequently.

## Step 4: Create a Loyalty Flag for Existing Customers

In [36]:
```
# Assuming 'ords_prods_merge' dataframe is already loaded

# Create a new column 'order_count' to store the number of unique orders for each user
ords_prods_merge['order_count'] = ords_prods_merge.groupby('user_id')['order_number'].transform('nunique')

# Define the criteria for loyalty categories based on 'order_count'
ords_prods_merge['loyalty_flag'] = 'New Customer'  # Default category
ords_prods_merge.loc[ords_prods_merge['order_count'] > 20, 'loyalty_flag'] = 'Loyal Customer'
ords_prods_merge.loc[(ords_prods_merge['order_count'] > 10) & (ords_prods_merge['order_count'] <= 20), 'loyalty_flag'] = 'Regular Customer'

# Display the distribution of loyalty categories
print(ords_prods_merge['loyalty_flag'].value_counts())
```

```
loyalty_flag
Loyal Customer      19349998
Regular Customer     6834798
New Customer         6249416
Name: count, dtype: int64
```

## Step 5: Analyze Spending Habits Based on Loyalty Category

In [39]:
```
# Calculate basic statistics of product prices for each loyalty category
price_stats_by_loyalty = ords_prods_merge.groupby('loyalty_flag')['prices'].describe()

# Display the result
print(price_stats_by_loyalty)
```

```
                       count       mean         std  min  25%  50%   75%  \
loyalty_flag
Loyal Customer    19349998.0  11.086957  419.748529  1.0  4.2  7.4  11.2
New Customer       6249416.0  13.294340  597.300832  1.0  4.2  7.4  11.3
Regular Customer   6834798.0  13.311951  582.885109  1.0  4.2  7.4  11.3

                      max
loyalty_flag
Loyal Customer    99999.0
New Customer      99999.0
Regular Customer  99999.0
```

## Analysis of Spending Habits by Loyalty Category

Analysis of Spending Habits by Loyalty Category The data reveals interesting patterns in spending across different loyalty categories:

Loyal Customers:

Average Spend: About 11 per purchase. Price Range: A wide variety of purchases, some reaching up to 99,999. Loyal Customers tend to show consistent spending patterns, with a mix of both low and high-priced items. New Customers:

Average Spend: Slightly higher, around $13. Price Range: Includes both low-cost and occasional high-ticket items. New Customers sometimes make significant purchases despite being newer to the platform. Regular Customers:

Average Spend: Comparable to New Customers, also around $13. Price Range: Broad, similar to other groups.ther groups.

## Step 6: Create a Spending Flag for Users

In [57]:
```
# Calculate the mean price of products purchased by each user
user_spending_mean = ords_prods_merge.groupby('user_id')['prices'].mean()

# Create a new column for spending categories
ords_prods_merge['spending_flag'] = ords_prods_merge['user_id'].map(user_spending_mean)

# Create a new column for spending categories as strings
ords_prods_merge['spending_category'] = 'Low spender'
ords_prods_merge.loc[ords_prods_merge['spending_flag'] >= 10, 'spending_category'] = 'High spender'

# Display the distribution of spending categories
print(ords_prods_merge['spending_category'].value_counts())
```

```
spending_category
Low spender     31798751
High spender      635461
Name: count, dtype: int64
```

## Step 7: Create an Order Frequency Flag

In [62]:
```
# Calculate the median of "days_since_prior_order" for each user
user_order_frequency = ords_prods_merge.groupby('user_id')['days_since_prior_order'].median()

# Create a new column for the numeric order frequency flag
ords_prods_merge['order_frequency_flag'] = ords_prods_merge['user_id'].map(user_order_frequency)

# Create a new column for the string labels
ords_prods_merge['order_frequency_category'] = 'Frequent customer'
ords_prods_merge.loc[ords_prods_merge['order_frequency_flag'] > 20, 'order_frequency_category'] = 'Non-frequent customer'
ords_prods_merge.loc[(ords_prods_merge['order_frequency_flag'] > 10) & (ords_prods_merge['order_frequency_flag'] <= 20), 'order_frequency_category'] = 'Regular customer'

# Display the distribution of order frequency categories
print(ords_prods_merge['order_frequency_category'].value_counts())
```

```
order_frequency_category
Frequent customer        21577409
Regular customer          7217134
Non-frequent customer     3639669
Name: count, dtype: int64
```

## 09. Export your dataframe as a pickle file and store it correctly in your 'Prepared Data' folder

In [65]:
```
# Check shape before exporting
ords_prods_merge.shape
```

Out[65]:
```
(32434212, 26)
```

In [75]:
```
import os

# Define the path where you want to save the file
path = r'C:\Users\Asus\Music\Instacart Basket Analysis\Data\Prepared Data'

# Ensure the directory exists
os.makedirs(path, exist_ok=True)

# Save the dataframe as a pickle file
ords_prods_merge.to_pickle(os.path.join(path, 'ords_prods_merge_final.pkl'))
```

In [ ]:
```
```