This script includes the following points:

In [61]:
```
orders_products_combined = pd.read_pickle('orders_products_combined.pkl')
```

In [63]:
```
print(orders_products_combined.shape)
```

```
(32434489, 11)
```

In [65]:
```
ords_prods_merge = pd.merge(orders_products_combined, df_prods, on='product_id', how='inner')
```

In [68]:
```
ords_prods_merge.to_csv('ords_prods_merge.pkl', compression='gzip')
```

CareerFoundry 4.7: Deriving New Variables

This script includes the following points:

Step 1: Complete Instructions for Creating "price_label" and "busiest_day" Columns

In [90]:
```
# Assume ords_prods_merge is already loaded from the previous steps
# Creating 'price_label' column
ords_prods_merge['price_label'] = ords_prods_merge['prices'].apply(lambda x: 'High' if x > 10 else ('Low' if x < 5 else 'Medium'))

# Creating 'busiest_day' column
# Count the number of orders per day
day_orders = ords_prods_merge['order_dow'].value_counts()

# Determine the busiest day
busiest_day = day_orders.idxmax()
ords_prods_merge['busiest_day'] = ords_prods_merge['order_dow'].apply(lambda x: 'Busiest day' if x == busiest_day else 'Other day')
```

Step 2: Modify "busiest_day" to "Busiest days" and Add "Slowest days"

In [93]:
```
# Sort the days by number of orders
sorted_day_orders = day_orders.sort_values(ascending=False)

# Identify the two busiest and two slowest days
busiest_days = sorted_day_orders.index[:2].tolist()
slowest_days = sorted_day_orders.index[-2:].tolist()

# Create the new column
ords_prods_merge['busiest_days'] = ords_prods_merge['order_dow'].apply(
    lambda x: 'Busiest days' if x in busiest_days else ('Slowest days' if x in slowest_days else 'Regular days')
)
```

Step 3: Check Values of the New Column for Accuracy

In [95]:
```
# Check the frequency of each category
print(ords_prods_merge['busiest_days'].value_counts())
```

```
busiest_days
Regular days    12927461
Busiest days    11875462
Slowest days     7631289
Name: count, dtype: int64
```

Step 4: Create "busiest_period_of_day" Column

Define periods of the day based on the frequency of orders:

Label periods with the most orders as "Most orders."

Label periods with average orders as "Average orders."

Label periods with the fewest orders as "Fewest orders."

In [99]:
```
# Count the number of orders per hour
hour_orders = ords_prods_merge['order_hour_of_day'].value_counts().sort_index()

# Define thresholds
most_orders_threshold = hour_orders.quantile(0.75)
fewest_orders_threshold = hour_orders.quantile(0.25)

# Create the column
ords_prods_merge['busiest_period_of_day'] = ords_prods_merge['order_hour_of_day'].apply(
    lambda x: 'Most orders' if hour_orders[x] >= most_orders_threshold else (
        'Fewest orders' if hour_orders[x] <= fewest_orders_threshold else 'Average orders'
    )
)
```

Step 5: Print the Frequency for the "busiest_period_of_day" Column

In [106…
```
# Check the frequency of each period
print(ords_prods_merge['busiest_period_of_day'].value_counts())
```

```
busiest_period_of_day
Most orders       16143132
Average orders    15694264
Fewest orders       596816
Name: count, dtype: int64
```

Step 6: Clean and Structure the Notebook

Ensure all code sections are well-commented.

Use clear section headings (# Creating price_label column).

Add markdown cells for explanations and observations.

Step 7. Export dataframe as a pickle file to "Script " folder.

In [120…
```
# Export the updated dataframe
ords_prods_merge.to_pickle(os.path.join(path, 'Data', 'Prepared Data', 'orders_products_merge__final.pkl'))
```

In [ ]:

In [ ]: