Exercise 4.4

Step 1: Loading the Dataset

First, we'll load the datasets from the provided file. Since the dataset seems to be provided in a CSV format, we'll use pandas to read it.

In [25]:
```
import pandas as pd
import numpy as np
import os
```

Set Path

In [8]:
```
path = r'C:\Users\Asus\Music\Instacart Basket Analysis'
```

In [12]:
```
# Import orders.csv using list
df_ords = pd.read_csv(os.path.join(path, 'Data', 'Original Data', 'orders.csv'), index_col = False)
```

In [16]:
```
# Import product.csv
df_prods = pd.read_csv(os.path.join(path, 'Data', 'Original Data', 'products.csv'), index_col = False)
```

In [18]:
```
# Importing data set departments.csv
df_dep = pd.read_csv(os.path.join(path, 'Data', 'Original Data', 'departments.csv'), index_col = False)
```

Step 2: Change an Identifier Variable's Format

identify an identifier variable that doesn't need to be numeric. From your dataset, the order_id seems to be a good candidate.

In [29]:
```
# Convert order_id to string
df_ords['order_id'] = df_ords['order_id'].astype(str)
```

Step 3: Change a Variable's Name

I changed the name of a variable with an unintuitive name. For instance, order_dow (day of the week) could be renamed to order_day_of_week.

In [32]:
```
# Rename the column
df_ords.rename(columns={'order_dow': 'order_day_of_week'}, inplace=False)
```

Out[32]:

	order_id	user_id	eval_set	order_number	order_day_of_week	order_hour_of_day	days_since_prior_order
0	2539329	1	prior	1	2	8	NaN
1	2398795	1	prior	2	3	7	15.0
2	473747	1	prior	3	3	12	21.0
3	2254736	1	prior	4	4	7	29.0
4	431534	1	prior	5	4	15	28.0
...	...	...	...	...	...	...	...
3421078	2266710	206209	prior	10	5	18	29.0
3421079	1854736	206209	prior	11	4	10	30.0
3421080	626363	206209	prior	12	1	12	18.0
3421081	2977660	206209	prior	13	1	12	7.0
3421082	272231	206209	train	14	6	14	30.0

3421083 rows × 7 columns

Step 4: Find the Busiest Hour for Placing Orders

I need to find the busiest hour for placing orders by examining the order_hour_of_day column.

In [37]:
```
# Find the frequency of each hour
busiest_hour = df_ords['order_hour_of_day'].mode()[0]
busiest_hour_count = df_ords['order_hour_of_day'].value_counts().max()

print(f"The busiest hour for placing orders is {busiest_hour} with {busiest_hour_count} orders.")
```

```
The busiest hour for placing orders is 10 with 288418 orders.
```

Determine the Meaning Behind a Value in df_prods

we need the df_prods dataframe and a data dictionary. Since it's not provided, I'll guide you on how to do it assuming you have the data.

In [46]:
```
# Step 5: Determine the Meaning Behind a Value in df_prods
# Example: Lookup the meaning of department_id = 4
# Assuming df_prods is my products dataframe and it has a 'department_id' column
department_id_meaning = df_prods[df_prods['department_id'] == 4]['department_id'].unique()

print(f"The meaning of department_id 4 is: {department_id_meaning}")
```

```
The meaning of department_id 4 is: [4]
```

Step 6: Create a Subset for Breakfast Items

Assuming breakfast items have a specific department_id (let's say it's 10 for example):

In [58]:
```
# Subset for breakfast items
breakfast_items = df_prods[df_prods['department_id'] == 10]
```

In [60]:
```
breakfast_items
```

Out[60]:

	product_id	product_name	aisle_id	department_id	prices
503	503	Wild Rice Blend	68	10	10.8
1000	1000	Apricots	18	10	12.9
5161	5161	Dried Mango	18	10	6.1
6194	6194	Organic Red Kidney Beans	68	10	14.1
6455	6455	Organic Magic Muesli	68	10	8.3
7314	7314	Organic Quick Rolled Oats	68	10	12.3
10224	10224	Organic Hunza Golden Raisins	18	10	5.4
10540	10540	Whole Medjool Dates	18	10	13.0
10915	10915	Organic Short Brown Sprouted Rice	68	10	7.7
11325	11325	Organic Hemp Plus Granola	68	10	12.4
12699	12699	Super Nutty Granola	68	10	4.5
14611	14611	Turkish Apricots	18	10	9.3
14665	14665	Organic White Popcorn	18	10	10.5
14985	14985	Organic Raspberry Muesli	68	10	12.5
19067	19066	Organic Wheat Bran	68	10	13.4
19629	19628	Organic Blueberries Package	18	10	2.7
22261	22260	Organic Rolled Oats	68	10	11.4
22828	22827	Organic Black Mission Figs	18	10	5.1
23039	23038	Organic Mung Beans	68	10	11.0
25860	25859	Cranberry Beans	68	10	6.7
27416	27414	Organic Emmer Farro	68	10	11.7
28069	28067	Organic Honey Lavender Granola	68	10	5.0
28657	28655	Crystallized Ginger Chunks	18	10	8.0
30367	30365	Vegetable Chips	18	10	7.2
32234	32232	Organic Roasted Buckwheat (Kasha)	68	10	13.6
38011	38007	Naturally Sweet Plantain Chips	18	10	1.4
38617	38613	Large Ataulfo Mango	68	10	5.6
39657	39653	Organic Split Green Peas	68	10	13.6
40401	40397	Organic Royal Rainbow Quinoa	68	10	10.8
42095	42091	Pesto Sauce	68	10	5.7
42138	42134	Israeli Couscous	68	10	5.0
43214	43210	Rolled Oats	68	10	5.1
43582	43578	Madagascar Pink Rice	68	10	3.8
43773	43769	Organic Pearled Barley	68	10	5.9
45686	45682	Organic Turkish Apricots	68	10	5.4
46893	46889	Organic Brown Basmati Rice	68	10	5.3
47493	47489	Organic Brown Jasmine Rice	68	10	5.6
48782	48778	Fit Super A Juice, Cold Pressed, Carrot/Apple/...	18	10	11.3

Step 7: Find Observations for Specific Departments

Assuming the department ids for alcohol, deli, beverages, and meat/seafood are known (let's use 1, 2, 3, 4 for example):

In [65]:
```
# Subset for specific departments
dinner_party_items = df_prods[df_prods['department_id'].isin([1, 2, 3, 4])]
```

Step 8: Count Rows in the Last DataFrame

In [70]:
```
# Count rows in the dinner_party_items dataframe
num_rows = dinner_party_items.shape[0]
print(f"The last dataframe has {num_rows} rows.")
```

```
The last dataframe has 7755 rows.
```

Step 9: Extract Information for User ID 1

In [75]:
```
# Extract information for user_id 1
user_1_data = df_ords[df_ords['user_id'] == 1]
print(user_1_data)
```

```
   order_id  user_id eval_set  order_number  order_dow  order_hour_of_day  \
0   2539329        1    prior             1          2                  8
1   2398795        1    prior             2          3                  7
2    473747        1    prior             3          3                 12
3   2254736        1    prior             4          4                  7
4    431534        1    prior             5          4                 15
5   3367565        1    prior             6          2                  7
6    550135        1    prior             7          1                  9
7   3108588        1    prior             8          1                 14
8   2295261        1    prior             9          1                 16
9   2550362        1    prior            10          4                  8
10  1187899        1    train            11          4                  8

    days_since_prior_order
0                      NaN
1                     15.0
2                     21.0
3                     29.0
4                     28.0
5                     19.0
6                     20.0
7                     14.0
8                      0.0
9                     30.0
10                    14.0
```

Step 10: Provide Basic Stats for User ID 1

In [78]:
```
# Basic stats for user_id 1
user_1_stats = user_1_data.describe()
print(user_1_stats)
```

```
       user_id  order_number  order_dow  order_hour_of_day  \
count     11.0     11.000000  11.000000          11.000000
mean       1.0      6.000000   2.636364          10.090909
std        0.0      3.316625   1.286291           3.477198
min        1.0      1.000000   1.000000           7.000000
25%        1.0      3.500000   1.500000           7.500000
50%        1.0      6.000000   3.000000           8.000000
75%        1.0      8.500000   4.000000          13.000000
max        1.0     11.000000   4.000000          16.000000

       days_since_prior_order
count                11.000000
mean                 19.000000
std                   9.038811
min                   0.000000
25%                  14.250000
50%                  19.500000
75%                  26.250000
max                  30.000000
```

Step 11: Check Organization and Structure

I am sure my notebook has clear section headings and comments explaining each step.

Step 12: Export df_ords as CSV

In [82]:
```
# # Export df_ords
df_ords.to_csv('orders_wrangled.csv', index=False)
```

Step 13: Export df_dep_t_new as CSV

In [89]:
```
# Export df_dep_t_new
df_dep.to_csv('departments_wrangled.csv', index=False)
```

In [ ]:
```

```