4.5 Data Consistency Checks

Step 1: Perform Consistency Checks on df_prods

In [8]:
```
# Import libraries
import pandas as pd
import numpy as np
import os
```

In [10]:
```
path = r'C:\Users\Asus\Music\Instacart Basket Analysis'
```

In [12]:
```
df_prods = pd.read_csv(os.path.join(path, 'Data', 'Original Data', 'products.csv'), index_col = False)
```

In [16]:
```
# Perform the consistency checks on df_prods:
# Check for missing values
missing_values_prods = df_prods.isnull().sum()
print(missing_values_prods)
```

```
product_id        0
product_name     16
aisle_id          0
department_id     0
prices            0
dtype: int64
```

In [18]:
```
# Check for duplicates
duplicate_prods = df_prods.duplicated().sum()
print(duplicate_prods)
```

```
5
```

In [20]:
```
# Check data types
data_types_prods = df_prods.dtypes
print(data_types_prods)
```

```
product_id        int64
product_name     object
aisle_id          int64
department_id     int64
prices          float64
dtype: object
```

step 2: Run df.describe() on df_ords

First, load my df_ords dataframe:

In [23]:
```
# Load the orders dataset
df_ords = pd.read_csv(os.path.join(path,'Data', 'Prepared Data', 'orders_wrangled.csv'), index_col = False)
```

In [25]:
```
# Run df.describe() on df_ords
describe_ords = df_ords.describe()
print(describe_ords)
```

```
          order_id       user_id  order_number     order_dow  \
count  3.421083e+06  3.421083e+06  3.421083e+06  3.421083e+06
mean   1.710542e+06  1.029782e+05  1.715486e+01  2.776219e+00
std    9.875817e+05  5.953372e+04  1.773316e+01  2.046829e+00
min    1.000000e+00  1.000000e+00  1.000000e+00  0.000000e+00
25%    8.552715e+05  5.139400e+04  5.000000e+00  1.000000e+00
50%    1.710542e+06  1.026890e+05  1.100000e+01  3.000000e+00
75%    2.565812e+06  1.543850e+05  2.300000e+01  5.000000e+00
max    3.421083e+06  2.062090e+05  1.000000e+02  6.000000e+00

       order_hour_of_day  days_since_prior_order
count       3.421083e+06            3.214874e+06
mean        1.345202e+01            1.111484e+01
std         4.226088e+00            9.206737e+00
min         0.000000e+00            0.000000e+00
25%         1.000000e+01            4.000000e+00
50%         1.300000e+01            7.000000e+00
75%         1.600000e+01            1.500000e+01
max         2.300000e+01            3.000000e+01
```

Step 3: Check for Mixed-Type Data in df_ords dataframe

In [28]:
```
# Check for mixed-type data
mixed_type_columns = df_ords.columns[df_ords.apply(lambda col: col.apply(type).nunique() > 1)]
print(mixed_type_columns)
```

```
Index([], dtype='object')
```

Step 4: Fix Mixed-Type Data

In my days_since_prior_order column have mixed types, convert them to a single type.

In [40]:
```
# Fix mixed-type data by converting all entries to strings (if applicable)
df_ords['days_since_prior_order'] = df_ords['days_since_prior_order'].astype(str)
```

check for missing values in your df_ords dataframe.

In [45]:
```
# Check for missing values
missing_values_ords = df_ords.isnull().sum()
print(missing_values_ords)
```

```
order_id                0
user_id                 0
eval_set                0
order_number            0
order_dow               0
order_hour_of_day       0
days_since_prior_order  0
dtype: int64
```

Report findings and propose an explanation:

Missing Values Report

The column days_since_prior_order has missing values. This could be because it's the first order placed by some users, so there's no prior order to calculate the days since.

Step 6: Address Missing Values

This method is to fill missing values with an appropriate value, such as 0 or the mean.

In [55]:
```
# Fill missing values with 0 (assuming first orders have no prior orders)
df_ords.loc[:, 'days_since_prior_order'] = df_ords['days_since_prior_order'].fillna(0)

# Verify that missing values are filled
print(df_ords)
```

```
         order_id  user_id eval_set  order_number  order_dow  \
0         2539329        1    prior             1          2
1         2398795        1    prior             2          3
2          473747        1    prior             3          3
3         2254736        1    prior             4          4
4          431534        1    prior             5          4
...           ...      ...      ...           ...        ...
3421078   2266710   206209    prior            10          5
3421079   1854736   206209    prior            11          4
3421080    626363   206209    prior            12          1
3421081   2977660   206209    prior            13          1
3421082    272231   206209    train            14          6

         order_hour_of_day days_since_prior_order
0                        8                    nan
1                        7                   15.0
2                       12                   21.0
3                        7                   29.0
4                       15                   28.0
...                    ...                    ...
3421078                 18                   29.0
3421079                 10                   30.0
3421080                 12                   18.0
3421081                 12                    7.0
3421082                 14                   30.0

[3421083 rows x 7 columns]
```

Explain the method:

Method Explanation

Missing values in days_since_prior_orderwere filled with 0 because it makes sense to assume that the first order has no prior order.

check for duplicate values in my df_ords data.

In [63]:
```
# Check for duplicate values
duplicate_ords = df_ords.duplicated().sum()
print(duplicate_ords)
```

```
0
```

Duplicate Values Report

the output of the script shows 0 duplicate values, it indicates that your dataframe does not contain any duplicate rows.

Step 8: Address Duplicate Values

In [67]:
```
# Remove duplicate values
df_ords.drop_duplicates(inplace=True)
```

Method Explanation

Duplicate values were removed to ensure each entry in the dataframe is unique, which is important for accurate analysis.

Step 9: Export the Final Cleaned Data

In [71]:
```
df_ords.to_csv(os.path.join(path, 'Data','Prepared Data', 'orders_cleaned.csv'))
df_prods.to_csv(os.path.join(path, 'Data','Prepared Data', 'products_cleaned.csv'))
```

In [ ]:
```

```