# Graduate Project: Retail Streaming Analytics and Market Basket Analysis

## Data Mining, Exploration and Discovery - Summer 2024


## Professor Stephen F. Elston

## By Edwin Tembo


## Executive Summary:


The goal of this project is the creation of a streaming analytics tool set to provide real-time insight on retail wholesale purchases. In doing so, organizations can closely monitor purchase activity from a more granular perspective as it unfolds daily.

The project has been modified from the proposal to include Market Basket Analysis instead of a Recommendation System.


## Value Proposition:

The value is derived from better inventory planning at higher levels of an organization. This will also aid in assuring that purchase activities are in line with organizational requirements and strategic initiatives. This could lead to potentially higher revenue and profitability as leaders gain access to data that could facilitate faster decision making, and better inventory planning, purchase and delivery decisions. Besides the potential for greater profitability, this could streamline an organization's purchasing processes and improve productivity for buying teams.

## Data Sources:

The data used for this project is the Iowa Liquor Retail Sales Dataset, a catalog of all wholesale orders of liquor by all grocery stores, liquor stores, convenience stores. The Data Schema is shown below. This data is queried from Google Cloud Platform's Big Query Data Warehouse. An explanation of the data can be found here: https://console.cloud.google.com/marketplace/product/iowa-department-of-commerce/iowa-liquor-sales

Although the data focuses on Liquor Products Wholesale, the same data mining methods can be used for any wholesale products.

## Methodology:

This data is based on daily sales. To frame this as a streaming problem, purchase times will be randomized throughout each date on which purchase data was recorded, assuming a fixed hours of operation for departments involved in purchasing to for each retailer. This idea also assumes that daily purchases are expected for delivery within a short period of time and can be applied to distribution systems that depend on such assumptions.

# The following steps are proposed:

## 1. Exploratory Data Analysis

- The data will be explored to find missing values and to validate the schema.
- Exploration will focus on thoroughly understanding the data.

## 2. **Simulation of timestamp of sale/purchase.**

- In this step, if no basis for a daily distribution is found through research, estimations will be made to randomly assign purchase times on each day. It may be the case that certain customers have more predictable purchase schedules within this dataset. However, we will frame this as a more flexible, randomized schedule to facilitate the proposed streaming pipeline. This project will focus on the retailers with the highest statewide wholesale purchases.

## 3.**Streaming Analytics**

- Streaming Analytics , Exponential Moving Averages and Delta-Encoding will be used where applicable to provide larger distributors with estimates of real-time sales counts and revenues. Additionally, forecasts will be made using Second or Third order exponentially weighted moving averages where applicable.

## 4. **Market Basket Analysis**

- This section has been modified from a recommendations system to Market Basket Analysis. This is can give insight on the corporations buying activiates by using Frequent itemsets and Association Models like the Apriori Algorithm.

## The notebook below shows the findings from the initial exploration and simulation of purchase times.

# Required Libraries and Authentication

In [ ]:
```
#Required for plotly charts as images
!pip install -U kaleido
!pip install statsmodels==0.14.2
```

```
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl.metadata (15 kB)
Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 8.9 MB/s eta 0:00:00
Installing collected packages: kaleido
Successfully installed kaleido-0.2.1
Requirement already satisfied: statsmodels==0.14.2 in /usr/local/lib/python3.10/dist-packages (0.14.2)
Requirement already satisfied: numpy>=1.22.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels==0.14.2) (1.26.4)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3.10/dist-packages (from statsmodels==0.14.2) (1.13.1)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels==0.14.2) (2.1.4)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels==0.14.2) (0.5.6)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels==0.14.2) (24.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels==0.14.2) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels==0.14.2) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels==0.14.2) (2024.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.6->statsmodels==0.14.2) (1.16.0)
```

In [ ]:
```
import gc
import os
import random
import json
import pandas as pd
import numpy  as np
import plotly.express as px
import plotly.io as pio
import plotly.graph_objects as go
from   google.colab import auth
import bigframes.pandas as bf
from   google.colab import userdata
from   IPython.display import Image
from   datetime import datetime, timedelta, date
```

```
import matplotlib.pyplot as plt
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt

auth.authenticate_user()
GCP_PROJECT = userdata.get("gcp_project")
!gcloud config set project {GCP_PROJECT}

import warnings
warnings.simplefilter("ignore", category=FutureWarning)

from statsmodels.tools.sm_exceptions import ConvergenceWarning
warnings.simplefilter("ignore", category=ConvergenceWarning)
```

```
Updated property [core/project].
```

# INITIAL EXPLORATION

The code below authenticates to a Google Cloud Platform Account and downloads the free Iowa Liquor Sales Dataset into a BigFrames Pandas DataFrame.

In [ ]:
```
bf.options.bigquery.location = "US"
bf.options.bigquery.project = GCP_PROJECT
df = bf.read_gbq("bigquery-public-data.iowa_liquor_sales.sales")
```

In [ ]:
```
#Data Columns
df.columns
```

Out[ ]:
```
Index(['invoice_and_item_number', 'date', 'store_number', 'store_name',
       'address', 'city', 'zip_code', 'store_location', 'county_number',
       'county', 'category', 'category_name', 'vendor_number', 'vendor_name',
       'item_number', 'item_description', 'pack', 'bottle_volume_ml',
       'state_bottle_cost', 'state_bottle_retail', 'bottles_sold',
       'sale_dollars', 'volume_sold_liters', 'volume_sold_gallons'],
      dtype='object')
```

In [ ]:
```
#Data Types
df.dtypes
```

Out[ ]:

| | 0 |
|---|---|
| **invoice_and_item_number** | string[pyarrow] |
| **date** | date32[day][pyarrow] |
| **store_number** | string[pyarrow] |
| **store_name** | string[pyarrow] |
| **address** | string[pyarrow] |
| **city** | string[pyarrow] |
| **zip_code** | string[pyarrow] |
| **store_location** | geometry |
| **county_number** | string[pyarrow] |
| **county** | string[pyarrow] |
| **category** | string[pyarrow] |
| **category_name** | string[pyarrow] |
| **vendor_number** | string[pyarrow] |
| **vendor_name** | string[pyarrow] |
| **item_number** | string[pyarrow] |
| **item_description** | string[pyarrow] |
| **pack** | Int64 |
| **bottle_volume_ml** | Int64 |
| **state_bottle_cost** | Float64 |
| **state_bottle_retail** | Float64 |
| **bottles_sold** | Int64 |
| **sale_dollars** | Float64 |
| **volume_sold_liters** | Float64 |
| **volume_sold_gallons** | Float64 |

**dtype:** object

Since the data is organized by date, it is critical to use only the data that's required. If the oldest date is outside or desired date window, then a filter can be applied accordingly. Since this is still exploratory, the minimum date will be set dyanmically, as a hyperparameter. The next section will expore some general statistics about the data.

In [ ]:
```
## Visualization of the first few rows
df.head()
```

Query job e7695f30-2a9f-4f6d-b78e-21a71f5a852d is DONE. 8.7 GB processed. Open Job
Query job 2824e31d-0929-4630-8a4b-d18201bb282c is DONE. 0 Bytes processed. Open Job
Query job d4ce90fd-2e51-4583-a5f5-47b92a331df6 is DONE. 1.6 kB processed. Open Job

Out[ ]:

| | invoice_and_item_number | date | store_number | store_name | address | city | zip_code | store_location | county_number | county | ... | item_number | item_description | pack | bottle_volume_ml | state_bottle_cost | state_bottle_re |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | INV-16108100138 | 2018-12-05 | 2594 | HY-VEE FOOD STORE / SIOUX CITY | 4500 SERGEANT ROAD | SIOUX CITY | 51106 | POINT (-96.34697 42.44740) | 97 | WOODBURY | ... | 57125 | CHI-CHI'S GOLD MARGARITA | 6 | 1750 | 6.17 | 9 |
| **1** | INV-26568700047 | 2020-04-14 | 2623 | HY-VEE FOOD STORE #4 / SIOUX CITY | 2827 HAMILTON BLVD | SIOUX CITY | 51104.0 | POINT (-96.41778 42.51989) | 97 | WOODBURY | ... | 18006 | BUFFALO TRACE BOURBON | 12 | 750 | 13.0 | 1 |
| **2** | INV-42803400010 | 2021-12-10 | 4741 | WALGREENS #03700 / COUNCIL BLUFFS | 535 E BROADWAY | COUNCIL BLUFFS | 51503.0 | POINT (-95.83970 41.26607) | 78 | POTTAWATTAMIE | ... | 37996 | SMIRNOFF 80PRF | 12 | 750 | 8.25 | 12 |
| **3** | S18056700002 | 2014-03-25 | 2604 | HY-VEE WINE AND SPIRITS / LEMARS | 1201 12TH AVE SW | LEMARS | 51031 | POINT (-96.18335 42.77826) | 75 | PLYMOUTH | ... | 11296 | CROWN ROYAL | 12 | 750 | 14.75 | 22 |
| **4** | INV-23970300034 | 2019-12-17 | 2592 | HY-VEE DRUGSTORE / MARSHALLTOWN | 1712 S CENTER ST | MARSHALLTOWN | 50158.0 | POINT (-92.91250 42.02753) | 64 | MARSHALL | ... | 1799 | CAPTAIN MORGAN ORIGINAL SPICED BARREL | 6 | 1750 | 18.0 | 2 |

5 rows × 24 columns

In [ ]:
```
#Analysis of Data Attributes
print("Number of Rows :",  len(df))
print("Minimum Date :", df.date.min() )
```

Query job ad51497a-b262-4dcf-80c8-bd13d36b336d is DONE. 0 Bytes processed. Open Job

```
Number of Rows : 29667436
```

Query job de5e75c8-260c-4938-9027-0ed4b6d7dd1b is DONE. 237.3 MB processed. Open Job

```
Minimum Date : 2012-01-03
```

Since this dataset is dynamic, we will limit it to only a few calendar years. In this case the filter returns any records included in the calenadar year (entire year) of the date that was exactly one year ago.

In [ ]:
```
lookback_yrs = 1

def get_start_date(lookback_yrs:int)->str :
  """
  Returns a start date in the past given the number of calendar years.
  Params:
  - lookback_yrs: int - The number of calendar years for the lookback period.
  """
  five_yrs = timedelta(days=365) * lookback_yrs
  dt = date.today()  - five_yrs
  start_date = f"{str(dt.year)}-01-01"
  return start_date

start_date = get_start_date(lookback_yrs)

start_date
```

Out[ ]:
```
'2023-01-01'
```

The next step filters the data based on the date to get the most relevant values and reduce the computational requirements.

In [ ]:
```
df_short = df[df["date"]>=start_date]
```

In [ ]:
```
df_short.head()
```

Query job 1caa0ba6-4506-4284-893d-dbf7ed94d851 is DONE. 8.7 GB processed. Open Job

Query job 89b172ed-67ea-49b0-bfbd-86c33a3e406e is DONE. 0 Bytes processed. Open Job

Query job 68117cd5-22dd-4dda-8d31-c3da36756049 is DONE. 1.5 kB processed. Open Job

Out[ ]:

| | invoice_and_item_number | date | store_number | store_name | address | city | zip_code | store_location | county_number | county | ... | item_number | item_description | pack | bottle_volume_ml | state_bottle_cost | state_bottle_retail | bottles_sol |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **5** | INV-71338600024 | 2024-06-18 | 4568 | SELECT MART / SIOUX CITY | 4103 FLOYD BLVD | SIOUX CITY | 51108 | POINT (-96.35867 42.54224) | <NA> | WOODBURY | ... | 86881 | SOUTHERN COMFORT MINI | 10 | 50 | 5.16 | 7.74 | 1 |
| **7** | INV-65670300133 | 2023-12-27 | 2670 | HY-VEE FOOD STORE / CORALVILLE | 2004 8TH ST | CORALVILLE | 52241.0 | POINT (-91.59203 41.68232) | <NA> | JOHNSON | ... | 67524 | KAHLUA COFFEE | 24 | 375 | 8.99 | 13.49 | |
| **9** | INV-64414400019 | 2023-11-17 | 2657 | HY-VEE FOOD STORE / BEDFORD | 1604 BENT | BEDFORD | 50833.0 | POINT (-94.72663 40.67622) | <NA> | TAYLOR | ... | 88018 | LUNAZUL BLANCO | 12 | 750 | 13.5 | 20.25 | |
| **11** | INV-65475400047 | 2023-12-20 | 2513 | HY-VEE FOOD STORE #2 (1285) / IOWA CITY | 812 S 1ST AVE | IOWA CITY | 52240.0 | POINT (-91.50090 41.65141) | <NA> | JOHNSON | ... | 36304 | HAWKEYE VODKA | 24 | 375 | 2.0 | 3.0 | 2 |
| **16** | INV-55178400011 | 2023-01-23 | 4749 | WALGREENS #05239 / DAVENPORT | 1660 W LOCUST ST | DAVENPORT | 52804 | POINT (-90.60027 41.53823) | <NA> | SCOTT | ... | 37993 | SMIRNOFF 80PRF | 48 | 200 | 2.54 | 3.81 | 1 |

5 rows × 24 columns

[5 rows x 24 columns in total]

In [ ]:
```
mem_usage = df_short.memory_usage().to_numpy()
print( "Earliest Date: " , df_short["date"].min())
print( "Number of Rows: ", len(df_short))
print("Total memory usage: " , mem_usage.sum()/(1024*1024*1024) , "GB")
```

Query job cef41531-d11c-4215-a35b-c64a01dfebf6 is DONE. 237.3 MB processed. Open Job

Query job 1bee5340-5d9f-48a9-aee5-eeb287f75831 is DONE. 237.3 MB processed. Open Job

Query job 1cc79961-1032-4237-aa2a-5fcf602689ce is DONE. 237.3 MB processed. Open Job

```
Earliest Date:  2023-01-02
Number of Rows:  4130621
Total memory usage:  0.8771024532616138 GB
```

**Converting to a Pandas DF:**

Since every query to a BigFrames DataFrame is a queued as a job, a faster, more efficient way to handle and explore the data is desireable. Now that the data is filtered and reduced to a manageable size that fits into memory, it can be loaded into a Pandas dataframe for faster processing.

In [ ]:
```
# Copy df_short locally then save to pandas
df_pd = df_short.to_pandas()
```

Query job eb50fc7b-c09b-4b9c-8bd1-fa63ddaf5a63 is DONE. 8.7 GB processed. Open Job

**Checking the date ranges:**

In [ ]:
```
df_pd.sort_values(by="date", inplace=True)
```

In [ ]:
```
#first 3 rows
df_pd.date[:3]
```

Out[ ]:

| | date |
|---|---|
| **44553** | 2023-01-02 |
| **47734** | 2023-01-02 |
| **403869** | 2023-01-02 |

**dtype:** date32[day][pyarrow]

In [ ]:
```
#last 3 rows
df_pd.date[-3:]
```

Out[ ]:

| | date |
|---|---|
| **29657372** | 2024-07-31 |
| **29659035** | 2024-07-31 |
| **29664919** | 2024-07-31 |

**dtype:** date32[day][pyarrow]

**Data Cleansing**

In [ ]:
```
## checking the max length of a zip code
def check_zip_len(data):
 return max(data.zip_code[data.zip_code.isnull() ==False].str.strip().str.len())

check_zip_len(df_pd)
```

Out[ ]:
```
7
```

It appears that some zip code data my have unwanted characters. We will examine this field.

In [ ]:
```
df_pd.zip_code[(df_pd.zip_code.astype(str).str.contains('[^0-9]', regex=True)) & (df_pd.zip_code.isnull()==False)].head()
```

Out[ ]:

| | zip_code |
|---|---|
| **2523183** | 50021.0 |
| **4532594** | 50266.0 |
| **10963537** | 50021.0 |
| **14267328** | 50266.0 |
| **23633717** | 50126.0 |

**dtype:** string

In [ ]:
```
## remove decimal points from zip_codes
df_pd.loc[df_pd.zip_code.isnull()==False, "zip_code"] =df_pd.loc[df_pd.zip_code.isnull()==False, "zip_code"].str.replace('\.[0-9]{0,}$', '', regex=True)
```

In [ ]:
```
#Check max zip length again
check_zip_len(df_pd)
```

Out[ ]:
```
5
```

At initial examination, duplicate values are dropped.

In [ ]:
```
#Reduction to relevant columns
df_pd.columns
```

Out[ ]:
```
Index(['invoice_and_item_number', 'date', 'store_number', 'store_name',
       'address', 'city', 'zip_code', 'store_location', 'county_number',
       'county', 'category', 'category_name', 'vendor_number', 'vendor_name',
       'item_number', 'item_description', 'pack', 'bottle_volume_ml',
       'state_bottle_cost', 'state_bottle_retail', 'bottles_sold',
       'sale_dollars', 'volume_sold_liters', 'volume_sold_gallons'],
      dtype='object')
```

In [ ]:
```
df_pd.drop_duplicates( inplace =True )
```

In [ ]:
```
#Drop rows with null values in all columns
df_pd.dropna(how="all", inplace=True)
```

# EXPLORATORY DATA ANALYSIS

This stage will handle missing values and further reduce the dataframe to only the relevant columns.

**Null values**

In [ ]:
```
# Loop to check for nulls
def check_null_rows(data):
  for column in data.columns:
    print(f"{column:25s}{data[column].isnull().sum()}")

check_null_rows(df_pd)
```

```
invoice_and_item_number  0
date                     0
store_number             0
store_name               0
address                  1194
city                     1194
zip_code                 1194
store_location           17936
county_number            4130621
county                   1194
category                 0
category_name            0
vendor_number            0
vendor_name              0
item_number              0
item_description         0
pack                     0
bottle_volume_ml         0
state_bottle_cost        0
state_bottle_retail      0
bottles_sold             0
sale_dollars             0
volume_sold_liters       0
volume_sold_gallons      0
```

Since almost all the rows are missing county_number, and a sizeable proportion are missing store_location, these columns can be removed as they won't be needed in our analysis.The county name column will be used to identify the geographic county instead. Store location contains longitude/latitude values that will not be necessary in this initial assessment.

In [ ]:
```
df_pd.drop(columns =["county_number", "store_location"], inplace=True)
```

Determining the ratio of sales data for rows with missing values will show their importance. First we will determine if the store number is unique for each store. In order to do so, we will use the store name and address.

In [ ]:
```
#A comparison of the store numbers missing the data#
county_check = df_pd.loc[df_pd.county.isnull()==True, "store_number"].unique()
city_check   = df_pd.loc[df_pd.city.isnull()==True, "store_number"].unique()
zip_check    = df_pd.loc[df_pd.zip_code.isnull()==True, "store_number"].unique()


if np.all(list(county_check) == list(city_check)):
  same_stores = np.all(list(county_check) ==list(zip_check))
else:
  same_stores = False

num_stores = len(set(list(county_check) + list(city_check) + list(zip_check)))
print("The same stores are responsible for the missing values: ", same_stores)
print("The number of stores responsible for missing data     : ", num_stores )
```

```
The same stores are responsible for the missing values:  True
The number of stores responsible for missing data     :  11
```

In this case, 11 store numbers are responsible for the missing data. We can now check if any rows for these store numbers have the data needed to impute the missing values.

In [ ]:
```
#stores with the same store number,  with the data needed for imputation
county_filter = (df_pd.store_number.isin(county_check)) & (df_pd.county.isnull() ==False)
county_source_stores = df_pd.loc[county_filter, ["store_number","store_name" , "county"]]

city_filter = (df_pd.store_number.isin(city_check)) & (df_pd.city.isnull() ==False)
city_source_stores = df_pd.loc[city_filter, ["store_number","store_name" , "city", "county"]]

zip_filter = (df_pd.store_number.isin(zip_check)) & (df_pd.zip_code.isnull() ==False)
```

```
zip_source_stores = df_pd.loc[zip_filter, ["store_number","store_name" , "zip_code"]]

print("# of store numbers found for missing county: ", len(county_source_stores.store_number.unique()) )
print("# of store numbers found for missing city:   " , len(city_source_stores.store_number.unique()))
print("# of store numbers found for missing zip_code :" , len(zip_source_stores.store_number.unique())   )
```

```
# of store numbers found for missing county:  11
# of store numbers found for missing city:    11
# of store numbers found for missing zip_code : 11
```

This section checks the uniqueness of the store_numbers that will be used for imputation of missing values.

In [ ]:
```
city_source_stores.groupby(["store_number","store_name" , "city"]).count().sort_values(by="store_number")
```

Out[ ]:

| | | | county |
|---|---|---|---|
| **store_number** | **store_name** | **city** | |
| **10094** | **CASEY'S #6134 / COUNCIL BLUFFS** | **COUNCIL BLUFFS** | 1737 |
| **10120** | **BP / WELTON** | **WELTON** | 460 |
| **10180** | **SWEETIES LIQUOR STORE AND COFFEE SHOP LLC / ODEBOLT** | **ODEBOLT** | 1373 |
| **10183** | **CROWN LIQUOR & SMOKE / IOWA CITY** | **IOWA CITY** | 1275 |
| **10270** | **LIQUOR LOVERS / ANKENY** | **ANKENY** | 4405 |
| **10395** | **MALIK'S #3 / MASON CITY** | **MASON CITY** | 944 |
| **2556** | **HY-VEE WINE AND SPIRITS (1170) / ESTHERVILLE** | **ESTHERVILLE** | 3553 |
| **3784** | **HARTIG DRUG #14 / INDEPENDENCE** | **INDEPENDENCE** | 2532 |
| **4498** | **SPEEDE SHOP / WINTHROP** | **WINTHROP** | 852 |
| **5436** | **SMOKIN' JOE'S #8 TOBACCO AND LIQUOR OUTLET** | **DAVENPORT** | 2697 |
| **6208** | **HY-VEE FAST AND FRESH / GRIMES** | **GRIMES** | 338 |

In [ ]:
```
zip_source_stores.groupby(["store_number","store_name" , "zip_code"]).count().sort_values(by="store_number")
```

Out[ ]:

| store_number | store_name | zip_code |
|---|---|---|
| 10094 | CASEY'S #6134 / COUNCIL BLUFFS | 51501 |
| 10120 | BP / WELTON | 52774 |
| 10180 | SWEETIES LIQUOR STORE AND COFFEE SHOP LLC / ODEBOLT | 51458 |
| 10183 | CROWN LIQUOR & SMOKE / IOWA CITY | 52246 |
| 10270 | LIQUOR LOVERS / ANKENY | 50023 |
| 10395 | MALIK'S #3 / MASON CITY | 50401 |
| 2556 | HY-VEE WINE AND SPIRITS (1170) / ESTHERVILLE | 51334 |
| 3784 | HARTIG DRUG #14 / INDEPENDENCE | 50644 |
| 4498 | SPEEDE SHOP / WINTHROP | 50682 |
| 5436 | SMOKIN' JOE'S #8 TOBACCO AND LIQUOR OUTLET | 52806 |
| 6208 | HY-VEE FAST AND FRESH / GRIMES | 50111 |

In [ ]:
```
county_source_stores.groupby(["store_number","store_name" , "county"]).count().sort_values(by="store_number")
```

Out[ ]:

| store_number | store_name | county |
|---|---|---|
| 10094 | CASEY'S #6134 / COUNCIL BLUFFS | POTTAWATTAMIE |
| 10120 | BP / WELTON | CLINTON |
| 10180 | SWEETIES LIQUOR STORE AND COFFEE SHOP LLC / ODEBOLT | SAC |
| 10183 | CROWN LIQUOR & SMOKE / IOWA CITY | JOHNSON |
| 10270 | LIQUOR LOVERS / ANKENY | POLK |
| 10395 | MALIK'S #3 / MASON CITY | CERRO GORDO |
| 2556 | HY-VEE WINE AND SPIRITS (1170) / ESTHERVILLE | EMMET |
| 3784 | HARTIG DRUG #14 / INDEPENDENCE | BUCHANAN |
| 4498 | SPEEDE SHOP / WINTHROP | BUCHANAN |
| 5436 | SMOKIN' JOE'S #8 TOBACCO AND LIQUOR OUTLET | SCOTT |
| 6208 | HY-VEE FAST AND FRESH / GRIMES | POLK |

It appears values belong to the same set of stores. These can now be used to impute the missing values.

In [ ]:
```
#For City Imputation
city_imputation_values = city_source_stores.groupby(["store_number","store_name" , "city"]).count()
for i , r  in city_imputation_values.iterrows():
  print(i[0], i[2])
  df_pd.loc[(df_pd.city.isnull()==True) & (df_pd.store_number==i[0]), "city"] = i[2]

#release from CPU RAM
print("\n---RAM cleanup------")
del city_imputation_values
gc.collect()
```

```
10094 COUNCIL BLUFFS
10120 WELTON
10180 ODEBOLT
10183 IOWA CITY
10270 ANKENY
10395 MASON CITY
2556 ESTHERVILLE
3784 INDEPENDENCE
4498 WINTHROP
5436 DAVENPORT
6208 GRIMES

---RAM cleanup------
```

Out[ ]:
```
31
```

In [ ]:
```
#For Zip Code Imputation
zip_imputation_values = zip_source_stores.groupby(["store_number","store_name" , "zip_code"]).count()
for i , r  in zip_imputation_values.iterrows():
    print(i[0], i[2])
    df_pd.loc[(df_pd.zip_code.isnull()==True) & (df_pd.store_number==i[0]), "zip_code"] = i[2]

#release from CPU RAM
print("\n---RAM cleanup------")
del zip_imputation_values
gc.collect()
```

```
10094 51501
10120 52774
10180 51458
10183 52246
10270 50023
10395 50401
2556 51334
3784 50644
4498 50682
5436 52806
6208 50111

---RAM cleanup------
```

Out[ ]:
```
0
```

In [ ]:
```
#For county imputation
county_imputation_values = county_source_stores.groupby(["store_number","store_name" , "county"]).count()
for i , r  in county_imputation_values.iterrows():
    print(i[0], i[2])
    df_pd.loc[(df_pd.county.isnull()==True) & (df_pd.store_number==i[0]), "county"] = i[2]

#release from CPU RAM
print("\n---RAM cleanup------")
del county_imputation_values
gc.collect()
```

```
10094 POTTAWATTAMIE
10120 CLINTON
10180 SAC
10183 JOHNSON
10270 POLK
10395 CERRO GORDO
2556 EMMET
3784 BUCHANAN
4498 BUCHANAN
5436 SCOTT
6208 POLK

---RAM cleanup------
0
```

Out[ ]:

We will now compare any other rows with missing data to the total for the state to assure that any removed rows do not distort the sample. A threshold can be set to assure that an alert is triggered if the ratio of missing data rows to rows of the entire dataset is too high.

In [ ]:
```
#Note, this is 0.5%, LESS THAN 1%, Can be adjusted.
missing_vals_threshold= 0.005
```

In [ ]:
```
# Columns of important numeric values are used  for the comparison

city_sales_for_nulls = df_pd.loc[df_pd.city.isnull()==True,    ["bottle_volume_ml", "bottles_sold", "sale_dollars"]].sum()
zip_sales_for_nulls = df_pd.loc[df_pd.zip_code.isnull()==True, ["bottle_volume_ml", "bottles_sold", "sale_dollars"]].sum()
county_sales_for_nulls = df_pd.loc[df_pd.county.isnull()==True,["bottle_volume_ml", "bottles_sold", "sale_dollars"]].sum()

names = ["city", "zip", "county"]
missing_vals_list = [city_sales_for_nulls, zip_sales_for_nulls, county_sales_for_nulls]
```

In [ ]:
```
state_totals = df_pd[["bottle_volume_ml", "bottles_sold", "sale_dollars"]].sum()
```

In [ ]:
```
for i, v in enumerate(missing_vals_list):
  missing_vals_ratios = v/state_totals

  breaching_threshold = len(missing_vals_ratios[missing_vals_ratios > missing_vals_threshold])
  print(f"{names[i]}  : {breaching_threshold} numeric columns > {missing_vals_threshold * 100} % row limit.")
```

```
city  : 0 numeric columns > 0.5 % row limit.
zip  : 0 numeric columns > 0.5 % row limit.
county  : 0 numeric columns > 0.5 % row limit.
```

Checking rows with missing data once more.

In [ ]:
```
check_null_rows(df_pd)
```

```
invoice_and_item_number  0
date                     0
store_number             0
store_name               0
address                  1194
city                     0
zip_code                 0
county                   0
category                 0
category_name            0
vendor_number            0
vendor_name              0
item_number              0
item_description         0
pack                     0
bottle_volume_ml         0
state_bottle_cost        0
state_bottle_retail      0
bottles_sold             0
sale_dollars             0
volume_sold_liters       0
volume_sold_gallons      0
```

In this case only the address columns in still missing data. We will remove the column as our analysis will be based on larger geographics areas.

In [ ]:
```
df_pd.drop(columns="address", inplace=True)
```

# Vendor/Distributor Exploration

In [ ]:
```
df_pd[["vendor_name", "vendor_number", "store_number"]].groupby(["vendor_name","vendor_number" ])["store_number"].nunique().sort_values(ascending=False)
```

Out[ ]:

| | | store_number |
|---|---|---|
| **vendor_name** | **vendor_number** | |
| **DIAGEO AMERICAS** | **260** | 2184 |
| **SAZERAC COMPANY INC** | **421** | 2184 |
| **HEAVEN HILL BRANDS** | **259** | 2163 |
| **BROWN FORMAN CORP.** | **85** | 2160 |
| **JIM BEAM BRANDS** | **65** | 2157 |
| **...** | **...** | ... |
| **SANS WINE & SPIRITS CO** | **843** | 1 |
| **DC BEVERAGE LLC** | **638** | 1 |
| **NORSEMAN DISTILLERY LLC** | **938** | 1 |
| **RIVER VALLEY ORCHARDS & WINERY** | **792** | 1 |
| **MOUNTAIN LAUREL SPIRITS LLC** | **624** | 1 |

268 rows × 1 columns

**dtype:** int64

The data shows 266 Vendors for the chosen period with SAZERAC COMPANY INC servicing the most retail establishments. We'll futher examine SAZERAC COMPANY INC.

# Retailer Exploration

This section examines the store_number's with the highest sales.

In [ ]:
```
df_pd[["store_name", "store_number", "sale_dollars", "bottles_sold"]].groupby(["store_name", "store_number" ]) \
.agg({"bottles_sold" :"sum", "sale_dollars":"sum"}) \
.reset_index().sort_values(by="sale_dollars", ascending=False).head(10)
```

Out[ ]:

| | store_name | store_number | bottles_sold | sale_dollars |
|---|---|---|---|---|
| **1106** | HY-VEE #3 / BDI / DES MOINES | 2633 | 1257744 | 24040289.16 |
| **719** | CENTRAL CITY 2 | 4829 | 1133426 | 21777880.93 |
| **35** | ANOTHER ROUND / DEWITT | 5916 | 543783 | 10792098.47 |
| **1275** | HY-VEE WINE AND SPIRITS #1 (1281) / IOWA CITY | 2512 | 608790 | 9586684.21 |
| **65** | BENZ DISTRIBUTING | 3773 | 425261 | 8259785.49 |
| **2179** | WALL TO WALL WINE AND SPIRITS / WEST DES MOINES | 6242 | 229762 | 6363000.3 |
| **1296** | I-80 LIQUOR / COUNCIL BLUFFS | 4312 | 325110 | 5898255.01 |
| **760** | COSTCO WHOLESALE #788 / WDM | 3814 | 235733 | 5748446.86 |
| **2208** | WILKIE LIQUORS | 5102 | 344122 | 5545602.96 |
| **1810** | SAM'S CLUB 6344 / WINDSOR HEIGHTS | 3420 | 238355 | 4917323.98 |

Hyvee Stores has the highest sales. The next section will focus on all stores in the Hyvee Chain. The filter will be based in the name.

# Daily Sales Simulation:

This section will simulate timestamps for daily purchases.

In [ ]:
```
df_hyvee = df_pd[df_pd.store_name.str.replace('[^A-Za-z]', '', regex=True).str.upper().str.contains("HYVEE", regex=False)]
```

In [ ]:
```
## Further filtering to the last week of June 2024
df_hyvee.tail(3)
```

Out[ ]:

| | invoice_and_item_number | date | store_number | store_name | city | zip_code | county | category | category_name | vendor_number | ... | item_number | item_description | pack | bottle_volume_ml | state_bottle_cost | state_bottle |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **29624725** | INV-72752400043 | 2024-07-31 | 2509 | HY-VEE / DRUGTOWN #1 (7020) / CEDAR RAPIDS | CEDAR RAPIDS | 52404 | LINN | 1081300.0 | AMERICAN CORDIALS & LIQUEURS | 421 | ... | 84187 | 99 CINNAMON | 48 | 100 | 1.0 | |
| **29639389** | INV-72754300062 | 2024-07-31 | 2544 | HY-VEE FOOD STORE (1403) / MARSHALLTOWN | MARSHALLTOWN | 50158 | MARSHALL | 1041100.0 | AMERICAN DRY GINS | 434 | ... | 31656 | PARAMOUNT GIN | 12 | 750 | 3.75 | |
| **29652672** | INV-72757000073 | 2024-07-31 | 2502 | HY-VEE WINE AND SPIRITS (1022) / ANKENY | ANKENY | 50021 | POLK | 1031100.0 | AMERICAN VODKAS | 380 | ... | 37665 | ROW VODKA | 6 | 1750 | 12.22 | |

3 rows × 21 columns

In [ ]:
```
df_hyvee.columns
```

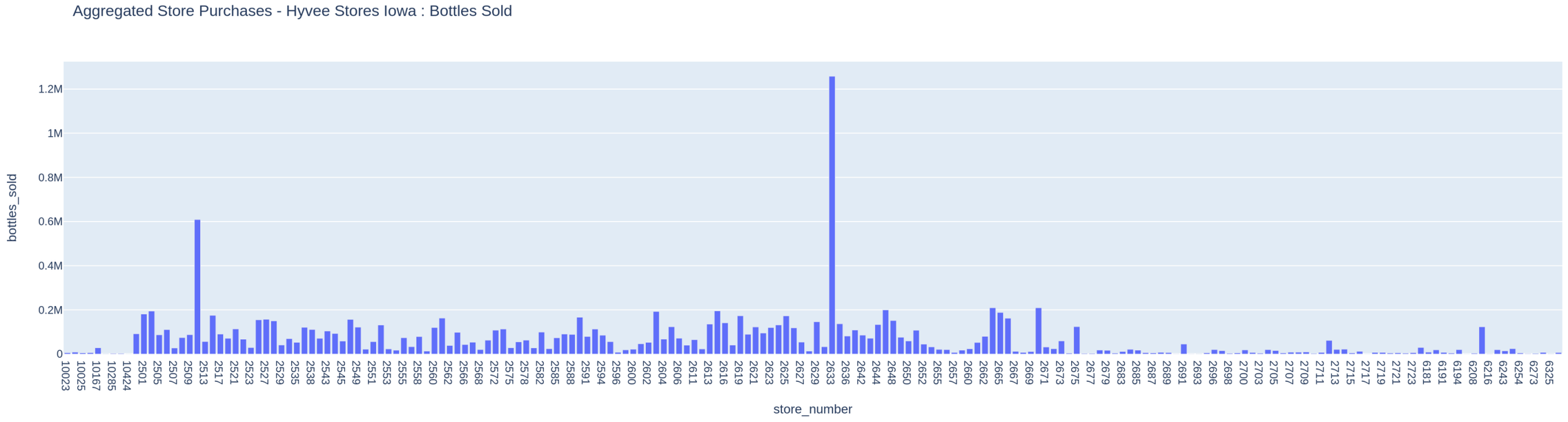
Out[ ]:
```
Index(['invoice_and_item_number', 'date', 'store_number', 'store_name', 'city',
       'zip_code', 'county', 'category', 'category_name', 'vendor_number',
       'vendor_name', 'item_number', 'item_description', 'pack',
       'bottle_volume_ml', 'state_bottle_cost', 'state_bottle_retail',
       'bottles_sold', 'sale_dollars', 'volume_sold_liters',
       'volume_sold_gallons'],
      dtype='object')
```

We'll determine examine aggregated store sales first.

In [ ]:
```
store_sales = df_hyvee.groupby(["store_number", "store_name", "city", "county"]).agg({"bottles_sold":"sum", "sale_dollars":"sum"}).reset_index()
```

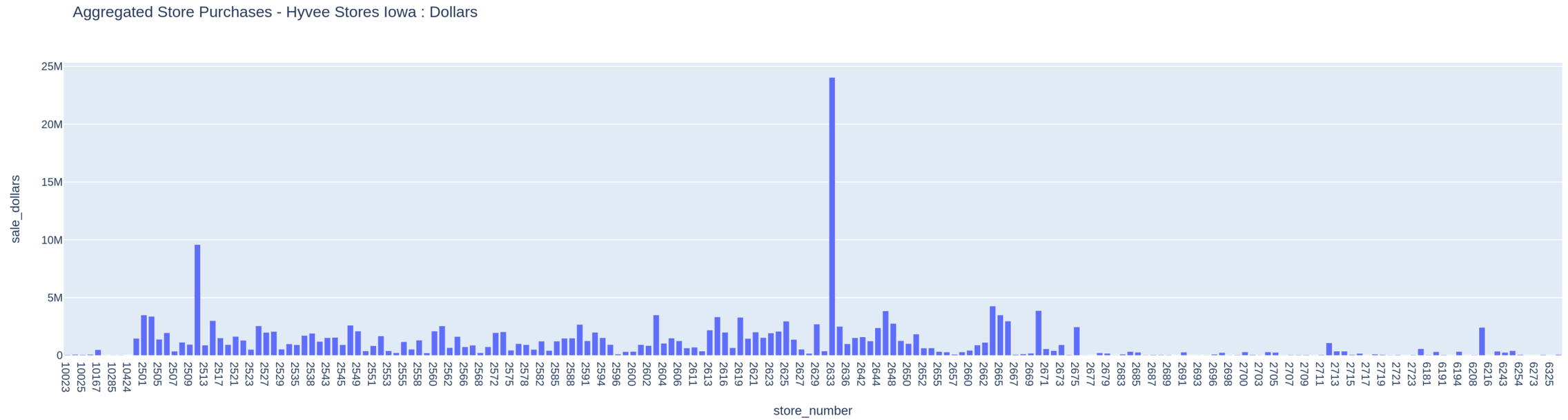
In [ ]:
```
fig = px.bar(store_sales, x="store_number", y="bottles_sold", title="Aggregated Store Purchases - Hyvee Stores Iowa : Bottles Sold")

Image(fig.to_image(format="png", width=1800, scale=2))
```

Out[ ]:




In [ ]:
```
fig = px.bar(store_sales, x="store_number", y="sale_dollars", title="Aggregated Store Purchases - Hyvee Stores Iowa : Dollars")
Image(fig.to_image(format="png", width =1800, scale = 2))
```

Out[ ]:




From interactive charts, Store number 2633 has the highest sales, both in terms of bottles bought. We will run our simulation tests on this store.

In [ ]:
```
df_hyvee_2633 = df_hyvee[df_hyvee.store_number == "2633"]
```

To get more insight into the store, we will examine a few more details.

In [ ]:
```
def avg_sales_df(data):
    avg_sales = data.groupby(["store_name", "item_description", "item_number", "pack"]).agg({"bottles_sold": "sum", "sale_dollars":"sum"}) \
    .reset_index() \
    .sort_values(by="bottles_sold", ascending=False)

    avg_sales.rename(columns = {"sale_dollars": "total_sale_dollars"}, inplace=True)
    avg_sales["dollars_per_bottle"] = round(avg_sales.total_sale_dollars/avg_sales.bottles_sold, 2)
    return avg_sales

avg_sales = avg_sales_df(df_hyvee_2633)
avg_sales
```

Out[ ]:

| | store_name | item_description | item_number | pack | bottles_sold | total_sale_dollars | dollars_per_bottle |
|---|---|---|---|---|---|---|---|
| **1738** | HY-VEE #3 / BDI / DES MOINES | TITOS HANDMADE VODKA | 38177 | 12 | 109668 | 2167039.68 | 19.76 |
| **1745** | HY-VEE #3 / BDI / DES MOINES | TORTILLA GOLD DSS | 77487 | 12 | 60873 | 447416.55 | 7.35 |
| **340** | HY-VEE #3 / BDI / DES MOINES | CAPTAIN MORGAN ORIGINAL SPICED | 43337 | 12 | 44832 | 857881.68 | 19.14 |
| **784** | HY-VEE #3 / BDI / DES MOINES | GOTCHA VODKA | 37258 | 12 | 40247 | 211296.75 | 5.25 |
| **989** | HY-VEE #3 / BDI / DES MOINES | JUAREZ GOLD | 89387 | 12 | 34356 | 439983.0 | 12.81 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **258** | HY-VEE #3 / BDI / DES MOINES | BP JEFFERSONS OCEAN WHEATED BOURBON BARREL | 19065 | 6 | 1 | 67.46 | 67.46 |
| **260** | HY-VEE #3 / BDI / DES MOINES | BP LAST DROP 1971 BLENDED WHISKY W/50ML | 100506 | 1 | 1 | 3000.0 | 3000.0 |
| **815** | HY-VEE #3 / BDI / DES MOINES | HA GLENFIDDICH 23YR GRAND CRU | 905896 | 3 | 1 | 270.0 | 270.0 |
| **814** | HY-VEE #3 / BDI / DES MOINES | HA ELMER T LEE SINGLE BARREL | 917946 | 12 | 1 | 32.25 | 32.25 |
| **937** | HY-VEE #3 / BDI / DES MOINES | JAMESON 18YR | 15639 | 3 | 1 | 164.99 | 164.99 |

1875 rows × 7 columns

The following section assumes that purchases occur at the retail store level, with a maximum of one purchase per day, per vendor. So, if a vendor supplies multiple products, all the product purchases for that day for that store, for that vendor are assumed to have occurred at the same time. However, purchases from different vendors to the same store can occur at different times on the same day. For simplicity, the smallest measure of time used to determine the timestamp is one minute.

In [ ]:
```
dates_per = df_hyvee.groupby(["vendor_number", "store_number","date"]).agg({"pack":"count"})
```

In [ ]:
```
dates_per = dates_per.reset_index().rename(columns = {"pack": "lines"}).sort_values(by=["lines"], ascending= False)
dates_per.head()
```

Out[ ]:

| | vendor_number | store_number | date | lines |
|---|---|---|---|---|
| **66425** | 260 | 2593 | 2024-04-09 | 78 |
| **68383** | 260 | 2623 | 2023-06-13 | 78 |
| **194610** | 421 | 2623 | 2023-06-13 | 77 |
| **194006** | 421 | 2616 | 2023-06-06 | 74 |
| **188302** | 421 | 2515 | 2023-03-09 | 72 |

In [ ]:
```
def sim_purchase_times(order_hrs = [8, 18]):
  num_mins = ((order_hrs[1] - order_hrs[0])  * 60)
  txn_time = datetime(1970, 1,1,8)+ timedelta(minutes = np.random.randint(1, num_mins))
  ## this assumes orders for each vendor at each store_number on each order date are submitted at the same time
  return txn_time.strftime("%H:%M")
```

In [ ]:
```
dates_per["purchase_ts"] = [sim_purchase_times() for i in range(0, len(dates_per))]
dates_per.head()
```

Out[ ]:

| | vendor_number | store_number | date | lines | purchase_ts |
|---|---|---|---|---|---|
| **66425** | 260 | 2593 | 2024-04-09 | 78 | 16:35 |
| **68383** | 260 | 2623 | 2023-06-13 | 78 | 17:50 |
| **194610** | 421 | 2623 | 2023-06-13 | 77 | 08:41 |
| **194006** | 421 | 2616 | 2023-06-06 | 74 | 09:41 |
| **188302** | 421 | 2515 | 2023-03-09 | 72 | 08:23 |

### Joining the Aggregated Purchase Time Data to the Full Hyvee DataFrame

In [ ]:
```
##checking for nulls in the aggregated dataset
check_null_rows(dates_per)
```

```
vendor_number            0
store_number             0
date                     0
lines                    0
purchase_ts              0
```

In [ ]:
```
initial_rows = len(df_hyvee)
```

In [ ]:
```
dates_per.drop(columns =["lines"], inplace=True)
```

In [ ]:
```
# This joins the two DataFrames and check for any missing values
df_hyvee = df_hyvee.merge(dates_per, how="left", on=["vendor_number", "store_number", "date"])
check_null_rows(df_hyvee)
```

```
invoice_and_item_number  0
date                     0
store_number             0
store_name               0
city                     0
zip_code                 0
county                   0
category                 0
category_name            0
vendor_number            0
vendor_name              0
item_number              0
item_description         0
pack                     0
bottle_volume_ml         0
state_bottle_cost        0
state_bottle_retail      0
bottles_sold             0
sale_dollars             0
volume_sold_liters       0
volume_sold_gallons      0
purchase_ts              0
```

In [ ]:
```
## row count validation
initial_rows ==len(df_hyvee)
```

Out[ ]:
```
True
```

In [ ]:
```
df_hyvee.dtypes
```

Out[ ]:

| | 0 |
|---|---|
| **invoice_and_item_number** | string[pyarrow] |
| **date** | date32[day][pyarrow] |
| **store_number** | string[pyarrow] |
| **store_name** | string[pyarrow] |
| **city** | string[pyarrow] |
| **zip_code** | string[pyarrow] |
| **county** | string[pyarrow] |
| **category** | string[pyarrow] |
| **category_name** | string[pyarrow] |
| **vendor_number** | string[pyarrow] |
| **vendor_name** | string[pyarrow] |
| **item_number** | string[pyarrow] |
| **item_description** | string[pyarrow] |
| **pack** | Int64 |
| **bottle_volume_ml** | Int64 |
| **state_bottle_cost** | Float64 |
| **state_bottle_retail** | Float64 |
| **bottles_sold** | Int64 |
| **sale_dollars** | Float64 |
| **volume_sold_liters** | Float64 |
| **volume_sold_gallons** | Float64 |
| **purchase_ts** | object |

**dtype:** object

In [ ]:
```
#Dropping extra rows
if ['level_0', 'index'] in list(df_hyvee.columns):
  df_hyvee.drop(columns = ['level_0', 'index'], inplace=True)
```

In the next step, the memory usage of the DataFrame is examined to make sure we can still work with Pandas.

In [ ]:
```
columns = [ "date", "purchase_ts","vendor_number", "store_number", "item_number", "bottles_sold", "sale_dollars", "city", "zip_code", "county"]
mem_estimation = sum(df_hyvee[columns].memory_usage(deep=True)) / (1024 * 1024 * 1024)
print("Memory usage : ", round(mem_estimation, 4), 'GB' )
```

```
Memory usage :  0.1453 GB
```

This is well within the available memory on the 12GB CPU RAM machine used for this project. At this point the total memory usage is 3.7 GB.

In [ ]:
```
## convert the date format
comparison_df = df_hyvee.copy()
comparison_df.drop(columns = [c for c in list(comparison_df.columns) if c not in columns], inplace=True)
```

```
if comparison_df.date.dtype == object:
  comparison_df["date"] = comparison_df["date"].astype(str).str.replace("\s{0,}[0-9]{2}\:[0-9]{2}\:[0-9]{2}\s{0,}$", '', regex=True)

comparison_df["txn_datetime"] = comparison_df["date"].astype(str).str.replace("\s{0,}[0-9]{2}\:[0-9]{2}\:[0-9]{2}\s{0,}$", '', regex=True)
comparison_df["txn_datetime"] = comparison_df["txn_datetime"].astype(str) + comparison_df["purchase_ts"].astype(str)
comparison_df["txn_datetime"] =  pd.to_datetime(comparison_df["txn_datetime"], format= "%Y-%m-%d%H:%M")
comparison_df.index = comparison_df["txn_datetime"]
comparison_df.drop(columns= "txn_datetime", inplace=True)
```

To check the results of the simulation times, the follwing section will order the DataFrame by Transaction Timestamp and store_number.

In [ ]:
```
comparison_df.sort_index(ascending= True, inplace=True)
comparison_df.head()
```

Out[ ]:

| txn_datetime | date | store_number | city | zip_code | county | vendor_number | item_number | bottles_sold | sale_dollars | purchase_ts |
|---|---|---|---|---|---|---|---|---|---|---|
| **2023-01-02 08:39:00** | 2023-01-02 | 2696 | DES MOINES | 50315 | POLK | 260 | 10804 | 10 | 75.0 | 08:39 |
| **2023-01-02 08:44:00** | 2023-01-02 | 2696 | DES MOINES | 50315 | POLK | 300 | 36901 | 6 | 42.3 | 08:44 |
| **2023-01-02 08:53:00** | 2023-01-02 | 2699 | ANKENY | 50023 | POLK | 260 | 43336 | 9 | 122.31 | 08:53 |
| **2023-01-02 08:53:00** | 2023-01-02 | 2699 | ANKENY | 50023 | POLK | 260 | 11296 | 3 | 78.72 | 08:53 |
| **2023-01-02 08:53:00** | 2023-01-02 | 2699 | ANKENY | 50023 | POLK | 260 | 10807 | 3 | 78.72 | 08:53 |

comparison_df[:100, :]

In [ ]:
```
date = '2024-01-03'
filter_date = pd.to_datetime(date, format='%Y-%m-%d')
fig = px.histogram(comparison_df.loc[comparison_df.date == filter_date ],
             x= comparison_df.loc[comparison_df.date == filter_date].reset_index().txn_datetime,
             y="bottles_sold",
             color="store_number",
             barmode="group",
             template=None,
             width   = 1500,
             title = f"Hyvee Stores: Simulated Iowa Liquor Wholesale Purchases By Timestamp per Store Number: {date} "
             )
Image(fig.to_image(format="png", width= 1200, height= 700,scale=2))
```

Out[ ]:


# Column Name Update

Because this project assumes that the users are working from the buyer's side, the names of some columns will be adjusted to prevent confusion.

In [ ]:
```
comparison_df.rename( columns = {"bottles_sold": "bottles_purchased","sale_dollars": "purchase_dollars"}, inplace=True)
```

In [ ]:
```
comparison_df.reset_index(inplace=True)
```

To help reduce the computation times when constructing the streams, we will give each order a ficticious order number, that is based on teh assumption that each store only submits a single order to each vendor on any given day. The datetime field is converted to Epochs and concatenated as a string to the store_number and vendor_number

In [ ]:
```
comparison_df["po_number"] = (comparison_df.store_number.astype(str) +
                              comparison_df.vendor_number.astype(str) +
                               (comparison_df.txn_datetime - pd.to_datetime('1970-01-01', format="%Y-%m-%d")).dt.total_seconds().astype(str) ).str \
                                .replace('[^0-9A-Za-z]', '', regex=True)
```

In [ ]:
```
comparison_df.head()
```

Out[ ]:

| | txn_datetime | date | store_number | city | zip_code | county | vendor_number | item_number | bottles_purchased | purchase_dollars | purchase_ts | po_number |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2023-01-02 08:39:00 | 2023-01-02 | 2696 | DES MOINES | 50315 | POLK | 260 | 10804 | 10 | 75.0 | 08:39 | 269626016726487400 |
| **1** | 2023-01-02 08:44:00 | 2023-01-02 | 2696 | DES MOINES | 50315 | POLK | 300 | 36901 | 6 | 42.3 | 08:44 | 269630016726490400 |
| **2** | 2023-01-02 08:53:00 | 2023-01-02 | 2699 | ANKENY | 50023 | POLK | 260 | 43336 | 9 | 122.31 | 08:53 | 269926016726495800 |
| **3** | 2023-01-02 08:53:00 | 2023-01-02 | 2699 | ANKENY | 50023 | POLK | 260 | 11296 | 3 | 78.72 | 08:53 | 269926016726495800 |
| **4** | 2023-01-02 08:53:00 | 2023-01-02 | 2699 | ANKENY | 50023 | POLK | 260 | 10807 | 3 | 78.72 | 08:53 | 269926016726495800 |

# Streaming Data:

To avoid the complexity of constructing a streaming data pipeline, this section will simply create some simple loop through data to simulate part of the retailer's central buying platform.

In [ ]:
```
filtered_cols = ['txn_datetime','date', 'store_number', 'vendor_number', 'item_number','bottles_purchased','purchase_dollars']

class PrepForStream:
  """
  This class aids in the creation of json files to send to streams
  """

  def __init__(self, in_data,start_date, end_date, columns=filtered_cols, exclude_returns=True):

    self.columns    = columns
    self.in_data    = in_data
    self.pre_df     = None
    self.out_json   = None
    self.start_date = start_date
    self.end_date   = end_date
    self.exclude_returns = exclude_returns


    if not isinstance(start_date,pd._libs.tslibs.timestamps.Timestamp ): start_date = pd.to_datetime(start_date, format='%Y-%m-%d')
    if not isinstance(end_date,pd._libs.tslibs.timestamps.Timestamp )  : end_date   = pd.to_datetime(end_date, format='%Y-%m-%d')

    date_filter  = (self.in_data.date >= start_date  ) & \
                   (self.in_data.date <= end_date)

    self.pre_df       = self.in_data.loc[date_filter,columns]

    if self.exclude_returns:
```

```
      self.pre_df  = self.pre_df[self.pre_df.bottles_purchased > 0 ]


  def gen_json_obj(self):
    self.pre_df_json   = self.pre_df.to_json(orient="records")


  def gen_json_output(self, store_numbers=None, vendor_numbers =None):
    """
       Send purchase data as JSON, One order Per Day , Per Vendor, Per Store

    """
    if self.pre_df is None:
      self.filter_date_range(self.start_date, self.end_date)

    store_numbers   = store_numbers if   store_numbers is not None  else self.in_data.store_number.unique()
    vendor_numbers  = vendor_numbers if  vendor_numbers is not None else self.in_data.vendor_number.unique()
    pre_filtered_df = self.pre_df.loc[(self.pre_df.store_number.isin(store_numbers)) & (self.pre_df.vendor_number.isin(vendor_numbers))]

    indices = pre_filtered_df.groupby(['txn_datetime', 'store_number', 'vendor_number']).agg({"date":"count"}).index

    self.out_json = [self.gen_json_obj(txn_datetime  = x[0],
                                       store_number  = x[1],
                                       vendor_number = x[2],
                                      )
                     for  x in indices

                    ]

start_stream = PrepForStream(in_data    = comparison_df
                            ,start_date = '2023-01-02',
                            end_date    = '2023-01-31'
                            )
```

# Analytics

# Delta Coding

This section introduces Delta Encoding as a way to reduce the storage requirements and consequently, the computational requirements of streaming analytics systems by computing aggregates and differences at longer intervals than the rate of incoming streams.

**The Kullback-Leibler Divergence**

The Kullback-Leibler Divergence is a commonly used Divergence metric that can be used to detect differences in probability distributions. Assuming a Corporate Buying department would like daily data captured when the daily distribution of total sku/item purchases differs from a set baseline known distribution, Delta Coding can be used to capture only specific data when the daily distributions of product purchases differ from the baseline. In the example below, the previous year's probability distribution for bottle counts per item_number/sku is used as the baseline. The implementation of the KL-Divergence is based on one used in SciPy at https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.kl_div.html#scipy.special.kl_div

In [ ]:
```
def get_items_count_distr(data,
                          groupby_col    = "item_number",
                          agg_dict       = {"bottles_purchased": "sum"} ,
                          rename_columns = {"bottles_purchased": "bottle_count"},
                          distr_col      = "bottle_count",
                          drop_reindexed = False):
    """
```

```
     Function to create a probability distribution for the counts
    """
    items = data.groupby(groupby_col).agg(agg_dict).rename(columns=rename_columns).reset_index()
    items[distr_col] = items.bottle_count/sum(items.bottle_count)
    return items

## For Data Range Filter
start = pd.to_datetime('2023-01-01', format='%Y-%m-%d')
end   = pd.to_datetime('2023-12-31', format='%Y-%m-%d')
date_filter = (comparison_df.date >= start) & (comparison_df.date <= end)

items_pdf_yr_2023 = get_items_count_distr(comparison_df[date_filter])
items_pdf_yr_2023
```

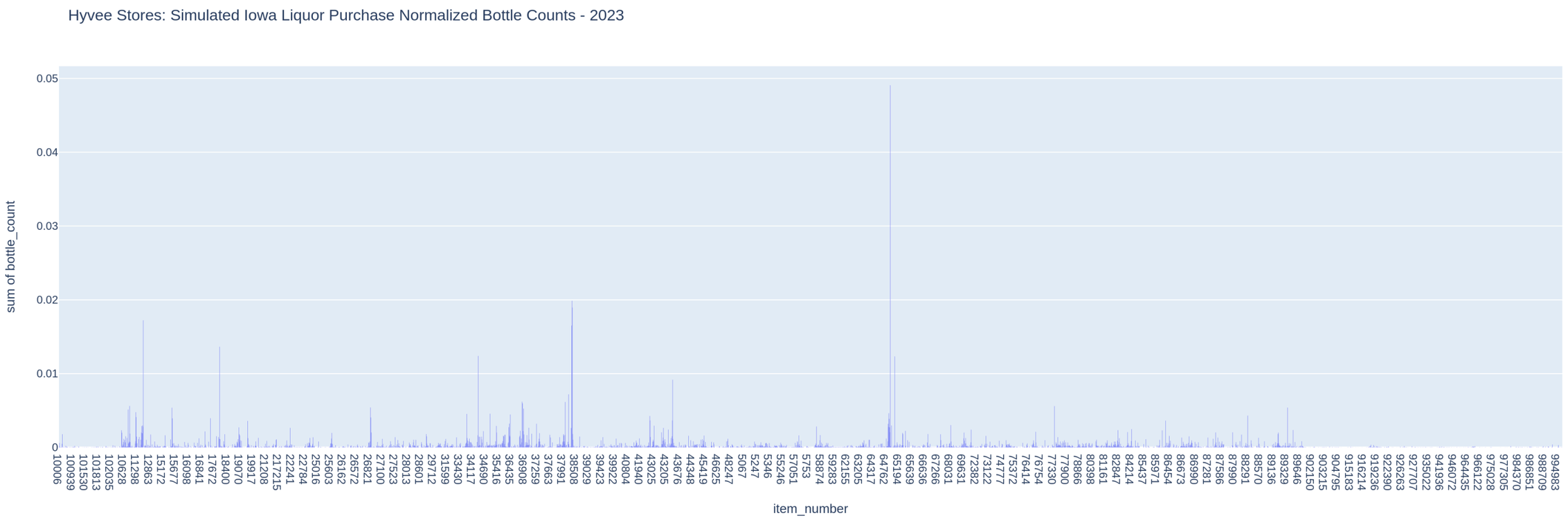
Out[ ]:

| | item_number | bottle_count |
|---|---|---|
| **0** | 10006 | 0.000114 |
| **1** | 10008 | 0.000606 |
| **2** | 10009 | 0.000076 |
| **3** | 100104 | 0.000019 |
| **4** | 100148 | 0.000024 |
| **...** | ... | ... |
| **4070** | 997874 | 0.000001 |
| **4071** | 999920 | 0.000001 |
| **4072** | 999927 | 0.000029 |
| **4073** | 999939 | 0.000009 |
| **4074** | 999940 | 0.000014 |

4075 rows × 2 columns

In [ ]:
```
from IPython.display import display
def plot_simple_histogram(data, x ,
                                y,
                          barmode,
                          template,
                          width,
                          title,
                          height=None,
                          as_image = True,
                          image_format="png",
                          image_width=1800,
                          image_scale=2.0 ):
    fig = px.histogram(data,
                       x= x,
                       y=y,
                       barmode =barmode,
                       template=template,
                       width   = width,
                       height  = height,
                       title =   title
                       )
    if as_image:
      display(Image(fig.to_image(format=image_format, width =image_width, scale = image_scale)))
```

```
    else:
      fig.show()


plot_simple_histogram(data   = items_pdf_yr_2023,
                      x      = "item_number",
                      y      ="bottle_count",
                      barmode="group",
                      template=None,
                      width   = 1800,
                      title   = f"Hyvee Stores: Simulated Iowa Liquor Purchase Normalized Bottle Counts - 2023" ,
                      height= 600,
                      as_image = True
)
```




The next section selects a random date after the last date used for the baseline distribution, converts the purchases to a distribution, then does a full outer join with the baseline distribution so that all products involved are included in the analysis.

In [ ]:
```
## Random Date in 2024 distribution
np.random.seed(19309)

def get_distrs(data,
               filter_by_date,
               baseline_data= items_pdf_yr_2023,
               feature_col  = "bottle_count",
               join_col     = "item_number"
                      ):
    """
      Function to combine Target Distribution with Baseline in a DataFrame
    """
```

```
    if filter_by_date is not None:
      tgt_data = data[data.date == filter_by_date]
    else:
      tgt_data = data
    tgt_data= get_items_count_distr(data = tgt_data)
    ## full outer  join using full item set to get all items from both the orig distr  and new one
    tgt_data = pd.merge(baseline_data[[join_col, feature_col]],tgt_data, how="outer" , on= join_col )
    tgt_data[[ f"{feature_col}_x", f"{feature_col}_y"]]= tgt_data[[f"{feature_col}_x", f"{feature_col}_y"]].fillna(0).astype(float)

    return  tgt_data

def get_random_date( data , end ):
    counter     = 0
    random_date = None
    record_count = (data.loc[max(list(data.index))][0] - end).days
    try_date    = True
    while try_date:
        random_idx = np.random.randint(0, record_count,  1)
        random_dates = data.date[data.date == end + pd.Timedelta(days=random_idx[0])].reset_index(drop=True)
        counter+=1
        if len(random_dates)> 0:
          random_date = random_dates.loc[0]
          try_date = False
        if counter >= 100:
          raise ValueError("Unabale to get date. Please try again.")

    return random_date



random_date = get_random_date(comparison_df, end = end)
combined_distrs = get_distrs(data           = comparison_df,
                             filter_by_date = random_date,
                             baseline_data= items_pdf_yr_2023,
                             feature_col  = "bottle_count",
                             join_col     = "item_number"
                            )

print(f"Target Date : {random_date}")

combined_distrs
```

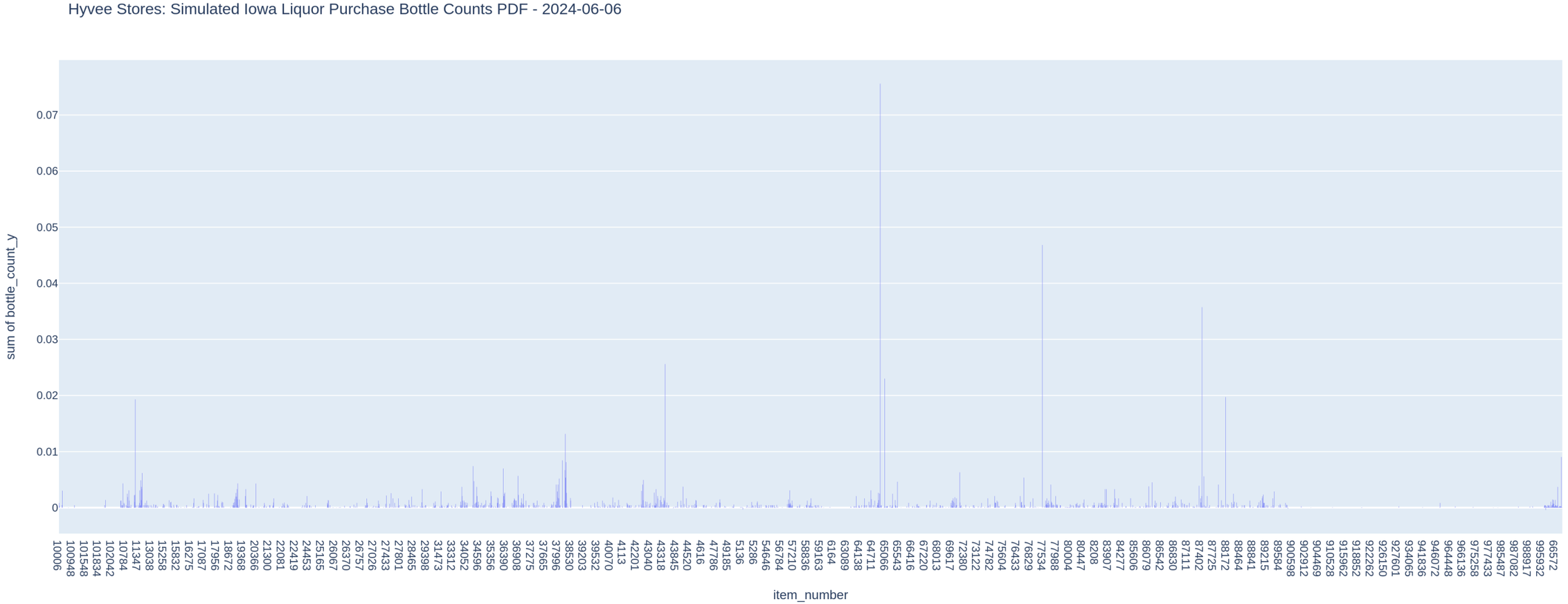
```
Target Date : 2024-06-06
```

Out[ ]:

| | item_number | bottle_count_x | bottle_count_y |
|---|---|---|---|
| **0** | 10006 | 0.000114 | 0.000821 |
| **1** | 10008 | 0.000606 | 0.000308 |
| **2** | 10009 | 0.000076 | 0.000000 |
| **3** | 100104 | 0.000019 | 0.000000 |
| **4** | 100148 | 0.000024 | 0.000000 |
| **...** | ... | ... | ... |
| **4120** | 87143 | 0.000000 | 0.000205 |
| **4121** | 88598 | 0.000000 | 0.000205 |
| **4122** | 916850 | 0.000000 | 0.009036 |
| **4123** | 921689 | 0.000000 | 0.000103 |
| **4124** | 928726 | 0.000000 | 0.000034 |

4125 rows × 3 columns

In [ ]:
```
plot_simple_histogram(data   = combined_distrs,
                      x      = "item_number",
                      y      = "bottle_count_y",
                      barmode="group",
                      template =None,
                      width    = 1800,
                      title    = f"Hyvee Stores: Simulated Iowa Liquor Purchase Bottle Counts PDF - {random_date}",
                      height   = 700,
                      as_image = True
)
```




The function below computes the KL Divergence based on the the Sci-Py Impelementation at https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.kl_div.html#scipy.special.kl_div .

In [ ]:
```
def kld_from_df(data, p_column, q_column ):
    """
      Calculate KL Divergence based on KLD(P||Q) = SUM ((P * log(P/Q))- P + Q) where P and Q are probablity distributions.
      This is based on the Implemenation from scipy.special
    """
    ## Set column to hold the inner log values.
    data["kld_inner"] = np.nan

    non_zero_filter = (data[q_column] > 0 ) & (data[p_column] > 0 )
    data.loc[non_zero_filter, "kld_inner"]  =np.log(data[p_column][non_zero_filter]/data[q_column][non_zero_filter] )

    zero_x_filter  =  (data[q_column] >= 0 ) & (data[p_column] == 0 )
    data.loc[zero_x_filter, "kld_inner"] = 0

    y_eq_zero_filter = data[q_column]==0
    data.loc[y_eq_zero_filter, "kld_inner"] = 0

    kld = np.sum((data[p_column] * data["kld_inner"]) - data[p_column] + data[q_column])

    return kld

kld_from_df(combined_distrs,"bottle_count_x", "bottle_count_y" ) ,kld_from_df(combined_distrs,"bottle_count_y", "bottle_count_x" )
```

Out[ ]: (0.2347877854075289, 0.7528522643820352)

In this section, the cumulative sums of the data per product are computed and saved. In doing so, only one data point per item will consume memory. This will also be an efficient process as the computation is a simple addition and hash table search or update for each product SKU/Item.

This will assume that failover protection and disaster recovery are implemented in a distributed system while using Pandas. Dask or Spark can be implemented to accomplish this.

In [ ]:
```
def run_stream_sim(data,
                   filter_by_date,
                   save_freq          =1000,
                   baseline_distr     = items_pdf_yr_2023,
                   kld_thresh         = [0.05, 0.9],
                   output_filepath    =f"./cumulative_sum_stream.csv"):
    """
      Helper function to simulate a stream. This will look though values.
      In real time, data would be passed in when available, but in this implementation the redundancy in processing
      multiple DataFrames is for simulation purposed.
    """
    date_idx          = np.argwhere(data.columns == 'date')[0][0]
    item_number_idx   = np.argwhere(data.columns == 'item_number')[0][0]
    qty_idx           = np.argwhere(data.columns == 'bottles_purchased')[0][0]

    columns =  ["last_row_id", "date" , "item_number", "bottles_purchased"]
    cumulative_sum_df = pd.DataFrame(columns= columns)
    counter = 0
    if filter_by_date is not None: data = data[data.date == filter_by_date]
    klds = []
    for i, row in data.iterrows():
      filter = cumulative_sum_df.item_number == row[item_number_idx]
      if len(cumulative_sum_df[filter] ) > 0 :
        cumulative_sum_df.loc[filter, "bottles_purchased"] = cumulative_sum_df.loc[filter, "bottles_purchased"] + row[qty_idx]
      else:
        cumulative_sum_df = pd.concat([cumulative_sum_df, pd.DataFrame( zip([i],[row[date_idx]], [row[item_number_idx]], [row[qty_idx]]), columns = columns)], axis=0)
        cumulative_sum_df.reset_index(inplace=True, drop=True)
      counter+=1

      ## save cumulative data at specific intervals
      if save_freq is not None:
         if counter%save_freq == 0 and counter > 0  :
          cumulative_sum_df.to_csv(output_filepath, header=True, index=False)


    #final save
    cumulative_sum_df.to_csv(output_filepath, header=True, index=False)

    ## KLD Values at end of period/date
    combed_distrs = get_distrs(cumulative_sum_df,
                               filter_by_date = filter_by_date,
                               baseline_data  = baseline_distr,
                               feature_col    = "bottle_count",
                                join_col       = "item_number"
                               )
    kld1= kld_from_df(combed_distrs,"bottle_count_x", "bottle_count_y" )
    kld2= kld_from_df(combed_distrs,"bottle_count_y", "bottle_count_x" )

    ##If thresholds breached return values
    if (kld1 < kld_thresh[0] or kld1 > kld_thresh[1] or kld2 < kld_thresh[0] or kld2 > kld_thresh[1]):
           klds.append([filter_by_date, kld1,kld2])
    if len(klds) > 0 :
      return klds
```

```
    return

#Lower and upper KL Divergence threshold values

thresh = [0.05, 0.9]
data = comparison_df
start_seed =52346
klds = []
for i in range(start_seed, start_seed+11):
    np.random.seed(i)
    random_date= get_random_date(data, end)
    print(random_date)
    kld_out = run_stream_sim(data, kld_thresh=thresh, filter_by_date = random_date)
    if kld_out is not None:
        klds.append(kld_out)
print("Dates With Purchases Breaching Thesholds:")
klds
```

```
2024-04-02
2024-03-25
2024-04-19
2024-02-15
2024-02-18
2024-07-23
2024-02-22
2024-02-22
2024-04-23
2024-07-16
2024-07-24
Dates With Purchases Breaching Thesholds:
```

Out[ ]:
```
[[[datetime.date(2024, 2, 18), -0.11971788084024397, 1.3806044443554613]]]
```

From the randomly selected dates, only one date was outside the selected threshold. Changing the threshold might result more dates selected.

This illustrates how Delta Coding can be used to report only the necessary data based on a threshold. The next section will attempt to find optimal thresholds using previous data. In this case, a DataFrame will be used to find KL Divergences for each given date between Jan-01 -2024 to June-30th-2024. The results will then be evaluated using data from July-2024.

In [ ]:
```
filter = (comparison_df.date > end) &   (comparison_df.date < pd.to_datetime('2024-07-01', format="%Y-%m-%d"))
daily_item_df = comparison_df[["date", "item_number", "bottles_purchased"]][filter].groupby(["item_number","date"]).agg({"bottles_purchased" :"sum"}).sort_values(by="date").reset_index()
daily_item_df.head(3)
```

Out[ ]:

| | item_number | date | bottles_purchased |
|---|---|---|---|
| **0** | 10805 | 2024-01-01 | 1 |
| **1** | 11296 | 2024-01-01 | 1 |
| **2** | 11776 | 2024-01-01 | 12 |

In [ ]:
```
klds_df      = pd.Series(daily_item_df.date.unique())
klds_df      = pd.DataFrame(klds_df, columns = ["date"])
klds_df.head(3)
```

Out[ ]:

| | date |
|---|---|
| **0** | 2024-01-01 |
| **1** | 2024-01-02 |
| **2** | 2024-01-03 |

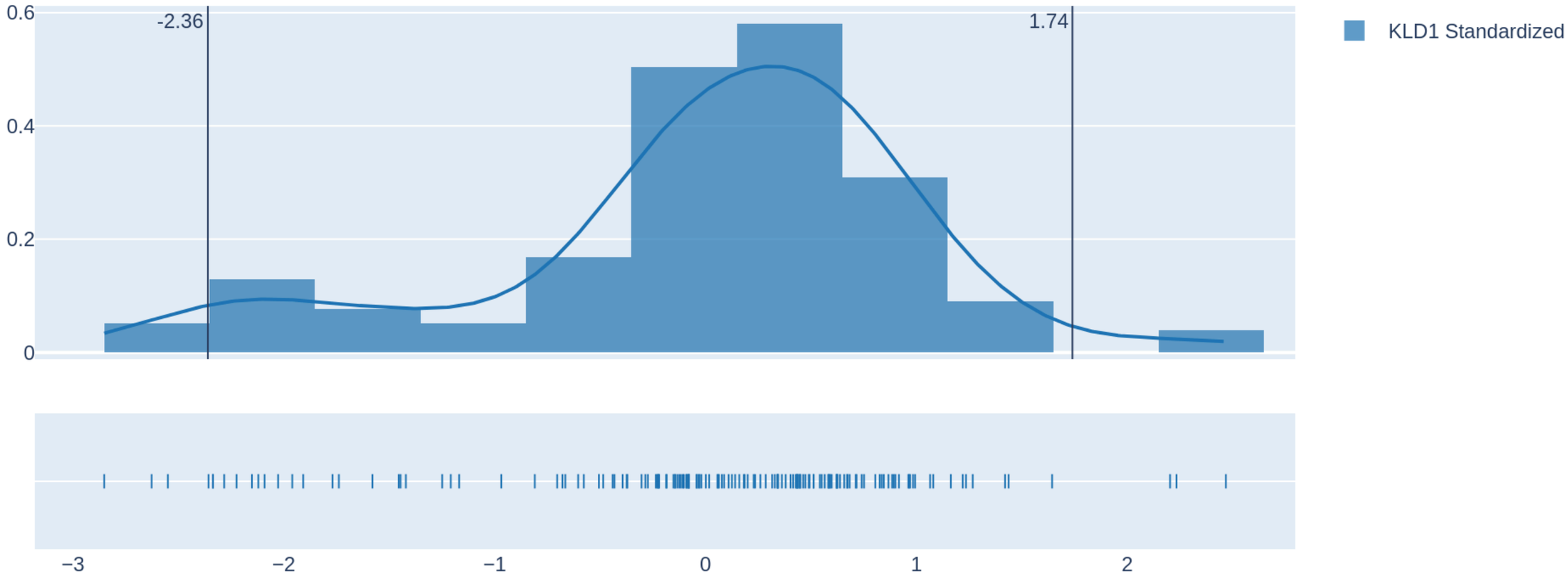
In [ ]:
```
def get_daily_kld(full_data, baseline_distr, filter_by_date):
    data_for_date = full_data[full_data["date"] ==filter_by_date]
    combed_distrs = get_distrs(data_for_date,
                               filter_by_date = filter_by_date,
                               baseline_data  = baseline_distr,
                               feature_col    = "bottle_count",
                               join_col       = "item_number"
                               )
    kld1= kld_from_df(combed_distrs,"bottle_count_x", "bottle_count_y" )
    kld2= kld_from_df(combed_distrs,"bottle_count_y", "bottle_count_x" )
    return [kld1,kld2]

klds_df[["kl_divergence1", "kl_divergence2"]] = [get_daily_kld(comparison_df, items_pdf_yr_2023, filter_by_date=d) for d in klds_df.date]
```

In [ ]:
```
#Standardization
klds_df[["kl_divergence1", "kl_divergence2"]].std().to_numpy()
klds_df["kl_divergence1_scaled"] = (klds_df["kl_divergence1"] - klds_df["kl_divergence1"].mean())/klds_df["kl_divergence1"].std()
klds_df["kl_divergence2_scaled"] = (klds_df["kl_divergence2"] - klds_df["kl_divergence2"].mean())/klds_df["kl_divergence2"].std()
```

To generate some thresholds, the known KL-Divergence data is standardized, and thresholds are set such that only outliers are reported. The thresholds are set towards the extremities of the standardized data. Visualizations will help find the best thresholds.

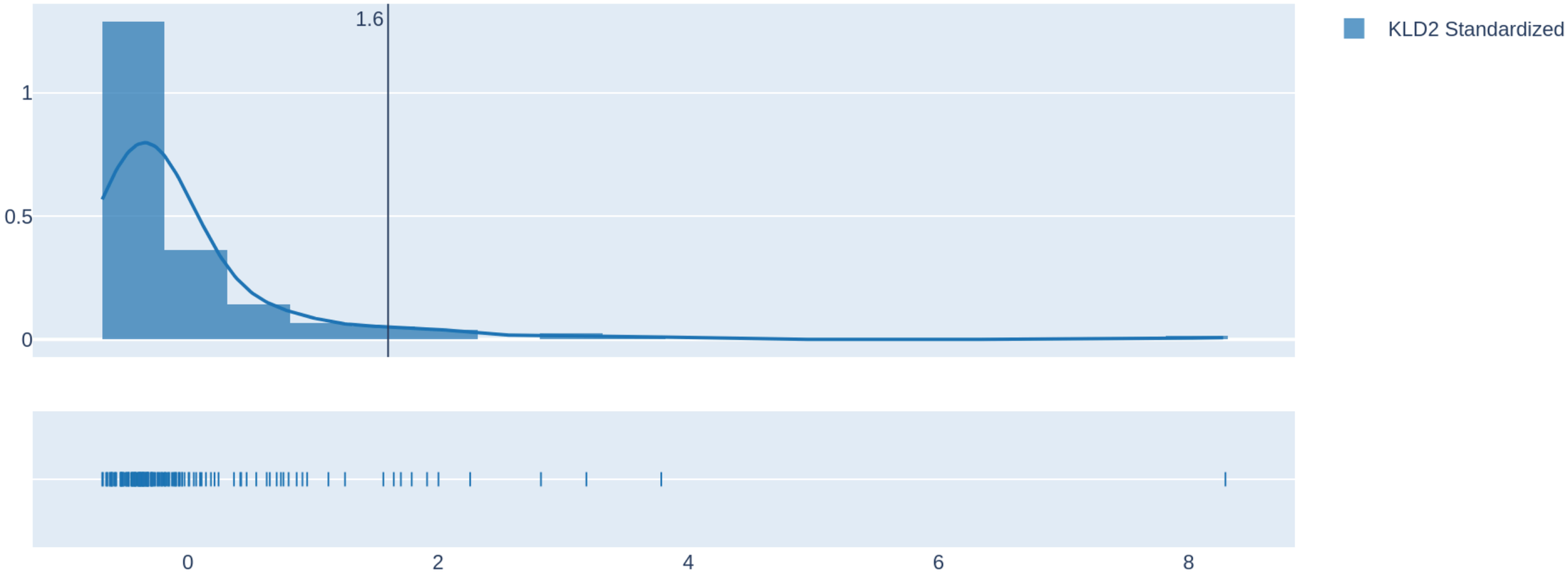
In [ ]:
```
import plotly.figure_factory as ff
import numpy as np
np.random.seed(1)

hist_data = [klds_df["kl_divergence1_scaled"]]
group_labels = ['KLD1 Standardized'] # name of the dataset


def pyplot_distplot(hist_data, group_labels, v_lines= None, as_image=True,bin_size=0.5, width=1000, scale=2.0):
    """
      Outputs a distribution plot with markers as vertical lines

    """
    fig = ff.create_distplot(hist_data, group_labels, bin_size=bin_size)

    for score in v_lines:
        fig.add_vline(x=score,
                      annotation_text=score, annotation_position="top left",
                      line_width=1)

    if not as_image:
      fig.show()

    else:

      return Image(fig.to_image(format="png", width = width, scale = scale))

v_lines = [1.74, -2.36]
pyplot_distplot(hist_data, group_labels,  v_lines= v_lines, as_image=True)
```

Out[ ]:




In [ ]:
```
hist_data = [klds_df["kl_divergence2_scaled"]]
group_labels = ['KLD2 Standardized'] # name of the dataset
v_lines = [1.6]
pyplot_distplot(hist_data, group_labels,  v_lines= v_lines, as_image=True)
```

Out[ ]:




The vertical lines show the outliers for the first KL-Divergence (KLD1) Line starting around 1.74 and -2.36 for the standardized values. The same can be seen for the KL-Divergence (KLD2) around 1.6. These standardized values will be used to set the thresholds for the actual KL Divergence values.

In [ ]:
```
kld1_thresh ={"lower" : klds_df.kl_divergence1[klds_df["kl_divergence1_scaled"] <= -2.36 ].max(), "upper": klds_df.kl_divergence1[klds_df["kl_divergence1_scaled"] >= 1.74 ].min()}

kld2_thresh ={"lower":None, "upper" :klds_df.kl_divergence2[klds_df["kl_divergence2_scaled"] >1.6 ].min() }

print(kld1_thresh, kld2_thresh)
```

```
{'lower': -0.19124183207998996, 'upper': 0.38481594333739505} {'lower': None, 'upper': 1.645240420736231}
```

While we can run the streaming simulation here, we will just use the data for July 2024 to see which would get reported if the data was getting streamed.

In [ ]:
```
df_July2024 = comparison_df["date"][ comparison_df.date >= pd.to_datetime('2024-07-01', format="%Y-%m-%d")].unique()
df_July2024 = pd.DataFrame(pd.Series(df_July2024) ,columns = ["date"])
df_July2024[["kl_divergence1", "kl_divergence2"]] = [get_daily_kld(comparison_df, items_pdf_yr_2023, filter_by_date=d) for d in df_July2024["date"]]

df_July2024.tail(3)
```

Out[ ]:

| | date | kl_divergence1 | kl_divergence2 |
|---|---|---|---|
| **26** | 2024-07-29 | 0.193665 | 0.818758 |
| **27** | 2024-07-30 | 0.230905 | 0.813413 |
| **28** | 2024-07-31 | 0.145448 | 0.537023 |

In [ ]:
```
filter1 = (df_July2024.kl_divergence1 < kld1_thresh.get("lower") ) | (df_July2024.kl_divergence1 >kld1_thresh.get("upper"))
filter2 = (df_July2024.kl_divergence2 < kld2_thresh.get("lower") ) | (df_July2024.kl_divergence2 > kld2_thresh.get("upper"))

df_July2024[filter1 | filter2]
```

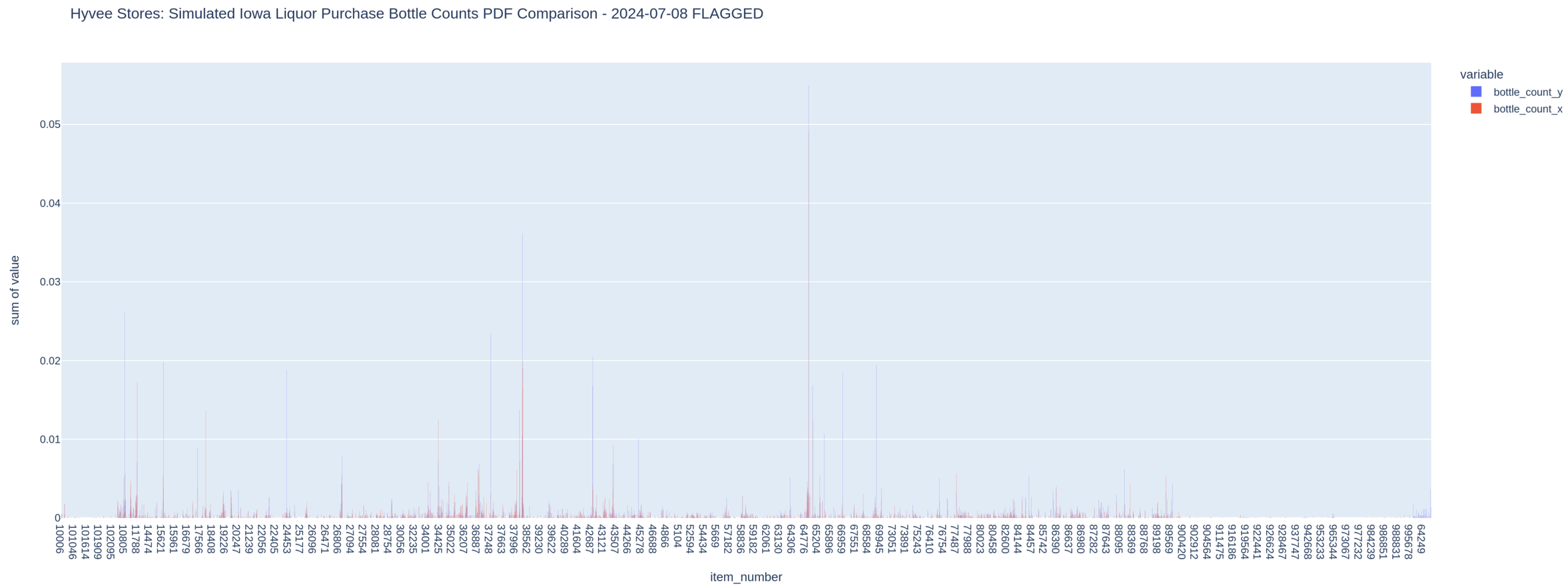
Out[ ]:

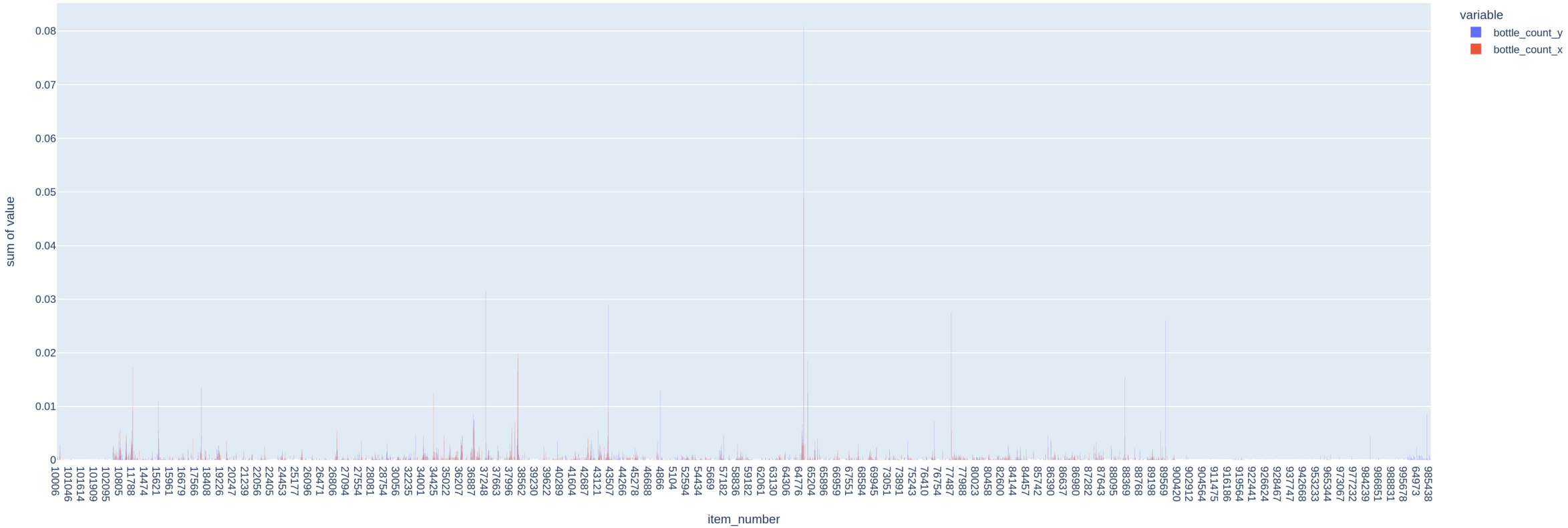
| | date | kl_divergence1 | kl_divergence2 |
|---|---|---|---|
| **6** | 2024-07-07 | 0.127999 | 1.719531 |
| **16** | 2024-07-18 | 0.444686 | 1.117852 |
| **25** | 2024-07-28 | -0.188729 | 2.422415 |

It appears three days in July-2024 had product purchase distributions outside the norm. A chart of one of these could provide more insight.

In [ ]:
```
date_val = df_July2024.loc[7][0]

combined_distrs = get_distrs(data           = comparison_df,
                             filter_by_date = date_val,
                             baseline_data  = items_pdf_yr_2023,
                             feature_col    = "bottle_count",
                             join_col       = "item_number"
                            )

plot_simple_histogram(data   = combined_distrs,
                      x      = "item_number",
                      y      = ["bottle_count_y", "bottle_count_x"],
                      barmode="group",
                      template =None,
                      width    = 1800,
                      title    = f"Hyvee Stores: Simulated Iowa Liquor Purchase Bottle Counts PDF Comparison - {date_val} FLAGGED",
                      height   = 700,
                      as_image = True
)

##
date_val = df_July2024.loc[21][0]
combined_distrs = get_distrs(data           = comparison_df,
                             filter_by_date = date_val,
                             baseline_data  = items_pdf_yr_2023,
                             feature_col    = "bottle_count",
                             join_col       = "item_number"
                            )
plot_simple_histogram(data   = combined_distrs,
                      x      = "item_number",
```

```
                    y      = ["bottle_count_y", "bottle_count_x"],
                    barmode="group",
                    template =None,
                    width    = 1800,
                    title    = f"Hyvee Stores: Simulated Iowa Liquor Purchase Bottle Counts PDF Comparison - {date_val} NOT FLAGGED",
                    height   =700,
                    as_image = True
)
```






**Conclusions**

The charts above show one of the three flagged dates (top chart) and a date that was not flagged for comparison. Each chart compares the baseline distribution(bottle_count_x) to the distribution for the target date (bottle_count_y). Some subtle differences are noticeable, perhaps some peaks that are larger than expected in the middle of the "FLAGGED" chart. This could indicate some unexpected buying activity that may need some review. The same system could be used in a real time streaming platform such that the flagged dates are saved. The thresholds can also be adjusted accordingly such that they reflect desired severity levels for reporting.

# Windowing Functions:

Windowing functions are another way to smooth streaming data. By specifying a window and stride parameter, data can be aggregated from streams in a memory and computationally efficient way. The Class below includes some Windowing Functions with options for further compression with Delta Coding, such that data is saved and processed only when certain events occur. The windowing parameters in this implementation can be based on both time and number of events.

In [ ]:
```
#Inherits the PrepForStream Class for general Data Processing
class StreamingAnalytics(PrepForStream):
  def __init__(self, in_data,start_date, end_date, columns=filtered_cols):
    """
    Class SteamingAnalytics inherits PrepFor Streams
    """

    super().__init__(in_data,start_date, end_date, columns)
```

```
  self.delta_params     = None
  self.use_delta_coding = False

def set_delta_coding(self, use_delta_coding):
    """
     Switch for delta coding
    """
    self.use_delta_coding = use_delta_coding


def set_delta_params(self , delta_params:list[dict]):
    """
    This function sets the delta coding params attribute
    Params:
        - columns        :   list[str]        - the list of columns with values to be filtered
        - threshold      :   list[int, float] - the value of the delta threshold
        - thresh_operator:   list[str]        - the operator >,<,>=,<=
        - filter_stats   :   list[str]        - mean
    """

    ## check shapes of inputs
    if not (  len(delta_params[0]) == 4):
      raise ValueError("Each entry must have the same number of values.")

    self.delta_params = delta_params


def window_stats(self,
                 window_size,
                 stride,
                 columns,
                 period_type = "time",
                 time_unit   = "min"):
"""
This is a windowing function to for smoothing a stream of data.
"""
if not time_unit in ['min', 'hr' 'day'] and period_type == "time":
  raise ValueError("Please use min, hr, or day for time_unit.")

counts = None
if period_type == 'time':
    ## start at the very begining of the day
    start = self.pre_df.txn_datetime.min()
    end   = start +  pd.Timedelta(window_size, time_unit)

    last_date      =  self.pre_df.txn_datetime.max().date()
    time_diff_mins = (self.pre_df.txn_datetime.max() - self.pre_df.txn_datetime.min())/pd.Timedelta(minutes=1)

    if time_unit   == 'min':
      time_diff_len   = time_diff_mins
    elif time_unit == 'hr':
      time_diff_len = time_diff_mins/60
    else:
      time_diff_len = time_diff_mins/(60 * 24)

    counts = (time_diff_len/stride ) if time_diff_len%stride == 0 else (int(time_diff_len/stride )) + 1
else:

    start = 0
    end = window_size
```

```
    counts = (len(self.pre_df)/stride ) if len(self.pre_df)%stride == 0 else (int(len(self.pre_df)/stride )) + 1

means_arr = None


for i in range(counts):


  if period_type == 'time':

      time_filter = (self.pre_df.txn_datetime >= start) & (self.pre_df.txn_datetime <= end)
      mean = self.pre_df.loc[time_filter, columns].mean()

  else:

      mean = self.pre_df.loc[start:end-1, columns].mean()

  mean = mean.fillna(0)
  ## add the timeframe
  if self.use_delta_coding:
    for param in self.delta_params:

       if param["filter_stat"] == "mean":
          pass_mean = False

          if param['thresh_op'] == '>' :

            pass_mean = True if mean.loc[param['column']] > param['threshold'] else False

          if param['thresh_op'] == '<'  :

            pass_mean = True if mean.loc[param['column']] < param['threshold'] else False

          if param['thresh_op'] == '>=' :

            pass_mean = True if mean.loc[param['column']] >= param['threshold'] else False

          if param['thresh_op'] == '<=' :

            pass_mean = True if mean.loc[param['column']] <= param['threshold'] else False

          if not pass_mean: mean = None

  if mean is not None:
      mean = np.insert(mean.to_numpy().reshape([1,len(columns)]) , 0 , end.strftime('%s') ,  axis= 1)
      mean = np.insert(mean , 0 , start.strftime('%s') ,  axis= 1)



  ## append the new aggs
  if means_arr is not None:
      means_arr = np.append(means_arr, mean, axis=0)  if mean is not None else means_arr
  else:
      means_arr = mean if mean is not None else means_arr


  ## set new start and end times
  start = start  + pd.Timedelta(stride, time_unit)       if period_type == 'time' else start + stride
  end   = start  + pd.Timedelta(window_size, time_unit)  if period_type == 'time' else start  + window_size

## output as DataFrames
df_columns = ["start", "end"] + columns if period_type != "time" else ["start_epoch", "end_epoch"] + columns
```

```
      means_out = pd.DataFrame(means_arr, columns =  df_columns)

      means_out[columns]=means_out[columns].fillna(0)


      del means_arr

      return means_out

streamer = StreamingAnalytics(in_data    = comparison_df
                             ,start_date = '2024-01-01',
                              end_date   = '2024-01-31'
)
```

In [ ]:
```
## Check for values less than zero. In this case they are nor necessary for this analysis
streamer.pre_df[streamer.pre_df.purchase_dollars <0]
```

Out[ ]:

| txn_datetime | date | store_number | vendor_number | item_number | bottles_purchased | purchase_dollars |
|---|---|---|---|---|---|---|

In [ ]:
```
streamer.pre_df.tail()
```

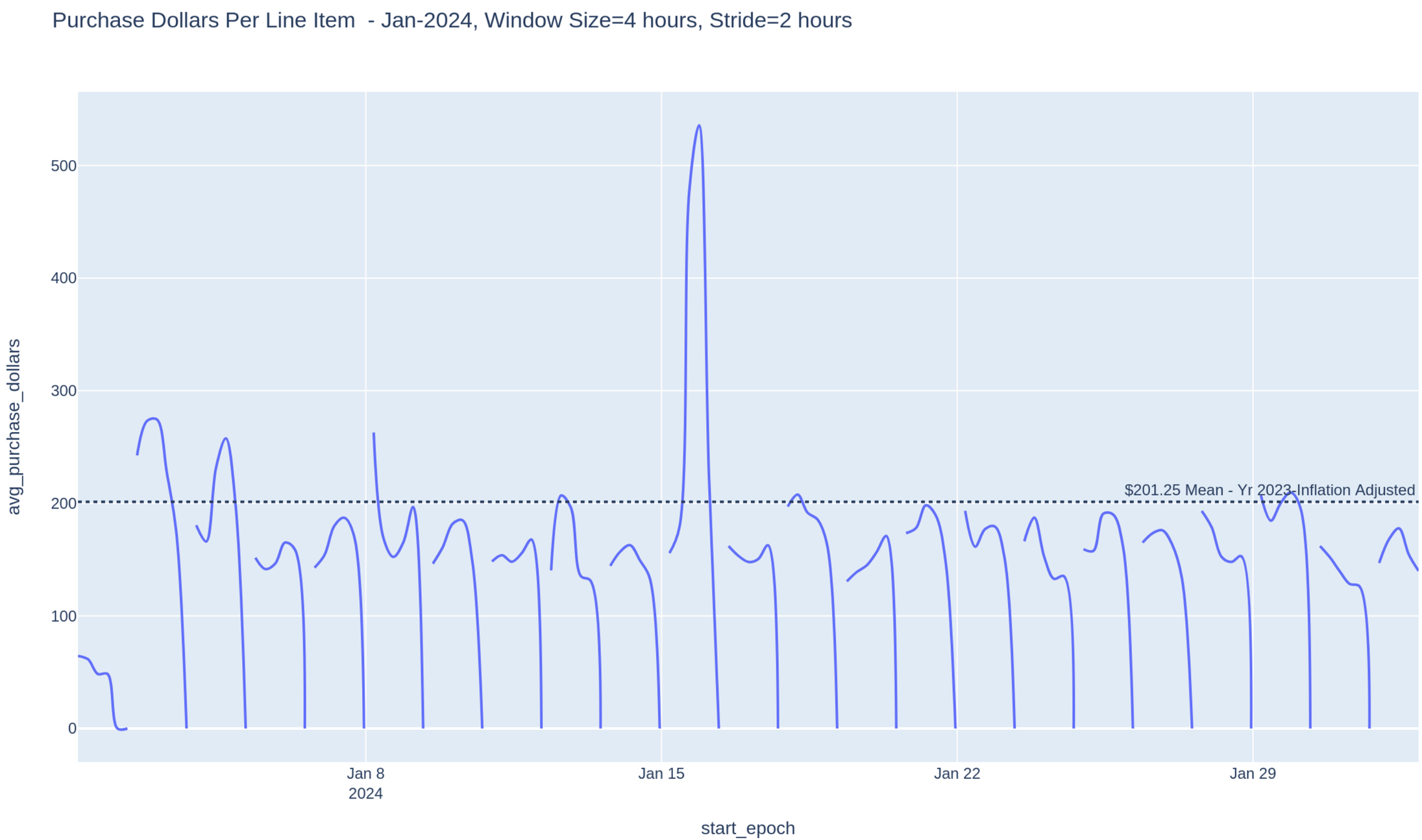
Out[ ]:

| | txn_datetime | date | store_number | vendor_number | item_number | bottles_purchased | purchase_dollars |
|---|---|---|---|---|---|---|---|
| **769194** | 2024-01-31 17:59:00 | 2024-01-31 | 2521 | 434 | 45278 | 6 | 72.0 |
| **769195** | 2024-01-31 17:59:00 | 2024-01-31 | 2521 | 434 | 41846 | 12 | 162.0 |
| **769196** | 2024-01-31 17:59:00 | 2024-01-31 | 2521 | 434 | 36305 | 12 | 63.0 |
| **769197** | 2024-01-31 17:59:00 | 2024-01-31 | 2521 | 434 | 45248 | 6 | 72.0 |
| **769198** | 2024-01-31 17:59:00 | 2024-01-31 | 2521 | 434 | 18201 | 6 | 121.5 |

In [ ]:
```
inflation_adj = 0.031
mean_yr_2023 = comparison_df.purchase_dollars.mean() + (comparison_df.purchase_dollars.mean() * inflation_adj)

means    = streamer.window_stats(stride      = 2*60,
                                 window_size = 4*60,
                                 period_type  ="time",
                                 columns= ["bottles_purchased", "purchase_dollars"]
                                )

means["start_epoch"] = pd.to_datetime(means["start_epoch"], unit="s")
means["end_epoch"]   = pd.to_datetime(means["end_epoch"]  , unit="s")
means.rename(columns = {"bottles_purchased" : "avg_bottles_purchased", "purchase_dollars" : "avg_purchase_dollars"}, inplace=True)
fig= px.line(means, x="start_epoch",
                    y="avg_purchase_dollars",
                    line_shape='spline',
                    title='Purchase Dollars Per Line Item  - Jan-2024, Window Size=4 hours, Stride=2 hours')
fig.update_xaxes(
        rangebreaks=[
            dict(bounds=["sat", "mon"]),
            dict(bounds=[19, 7], pattern="hour"), #hide non-business hours

        ]
    )

fig.add_hline(y= mean_yr_2023, line_dash="dot",
```

```
                annotation_text=f"${round(mean_yr_2023, 2)} Mean - Yr 2023-Inflation Adjusted",
                annotation_position="top right")

Image(fig.to_image(format="png", width= 1200, height= 700,scale=2))
```

Out[ ]:




The Windowing Function Above is compared to the previous year's inflation adjusted average. This can be used as an indication of an anomaly in any of the 4 hour periods. A few of the 4 hour periods exceed that threshold. The next section uses delta encoding to collect only the 4 hour windows with averages that exceed the threshold.

**Window Function with Delta Encoding:**

The following implemetation only reports data when the mean of the window exceeds a set threshold. This could be useful if teams only needs updates when buying activity for any window/rolling period exceeds or dips below set Threshold .

In [ ]:
```
#Enable Delta Encoding
delta_thresh = mean_yr_2023 + (0.2 *mean_yr_2023)
delta_params = [{"filter_stat" :"mean",
                 "column" :"purchase_dollars",
                 "thresh_op" :">" ,
                 "threshold" : delta_thresh }
               ]

streamer = StreamingAnalytics(in_data    = comparison_df
                             ,start_date = '2024-01-01',
                              end_date   = '2024-01-31'
)

streamer.set_delta_coding(use_delta_coding =True)
streamer.set_delta_params(delta_params)

means_w_delta  = streamer.window_stats(stride = 2*60,
                                       window_size = 4*60,
                                       period_type="time",
                                      columns= ["bottles_purchased", "purchase_dollars"]
                                )
means_w_delta.head(30)
```

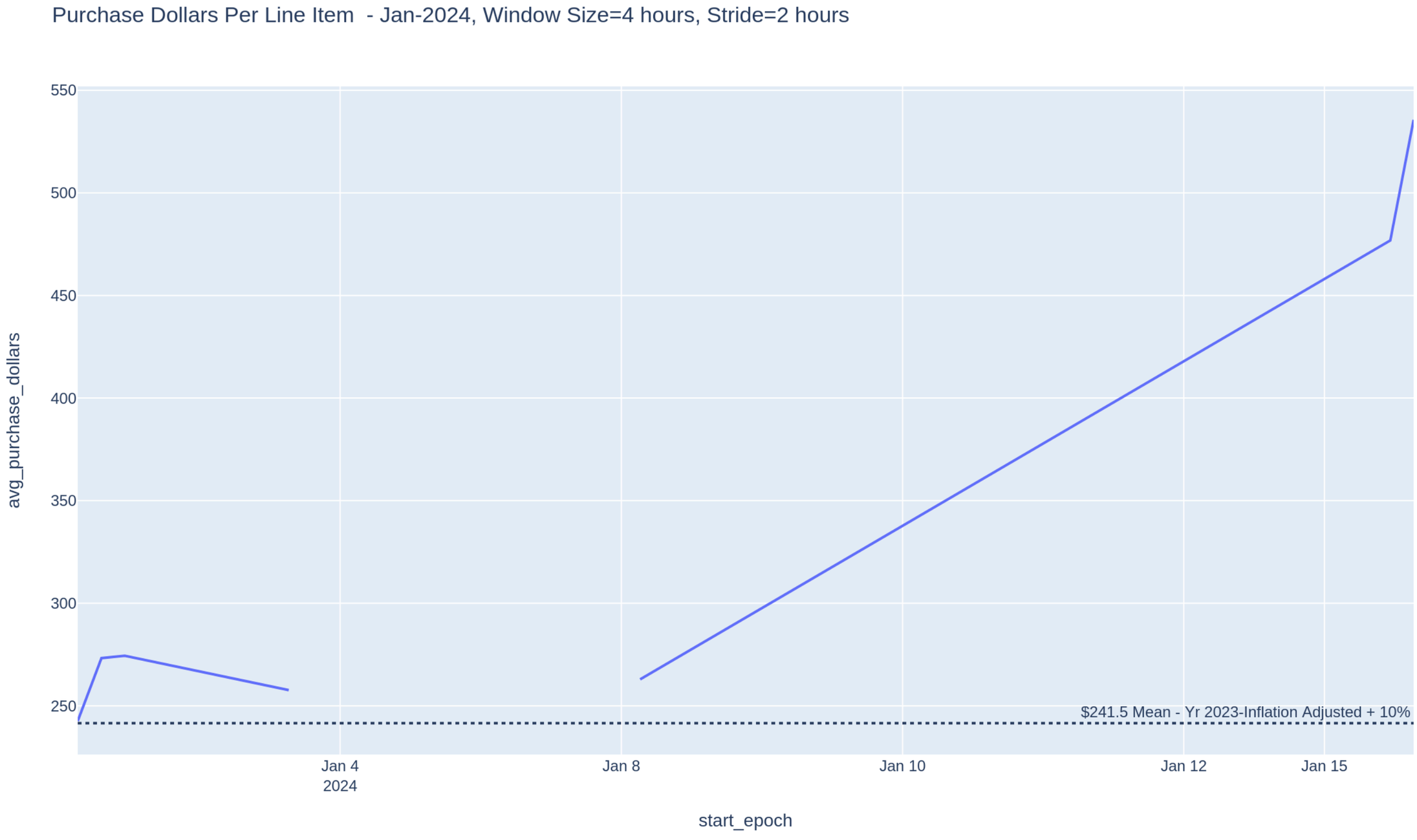
Out[ ]:

| | start_epoch | end_epoch | bottles_purchased | purchase_dollars |
|---|---|---|---|---|
| **0** | 1704184620 | 1704199020 | 14.334465 | 242.544304 |
| **1** | 1704191820 | 1704206220 | 14.546610 | 273.235331 |
| **2** | 1704199020 | 1704213420 | 14.843718 | 274.401246 |
| **3** | 1704292620 | 1704307020 | 15.245455 | 257.632530 |
| **4** | 1704695820 | 1704710220 | 20.337001 | 311.355034 |
| **5** | 1704703020 | 1704717420 | 16.630222 | 262.867218 |
| **6** | 1705322220 | 1705336620 | 26.930683 | 476.789990 |
| **7** | 1705329420 | 1705343820 | 30.735849 | 535.601380 |
| **8** | 1705984620 | 1705999020 | 17.529412 | 262.944853 |

In [ ]:
```
np.random.seed(1)
means_w_delta   = streamer.window_stats(stride      = 2*60,
                                        window_size = 4*60,
                                        period_type  ="time",
                                        columns= ["bottles_purchased", "purchase_dollars"]
                                )

means_w_delta["start_epoch"] = pd.to_datetime(means_w_delta["start_epoch"], unit="s")
means_w_delta["end_epoch"]   = pd.to_datetime(means_w_delta["end_epoch"]  , unit="s")
means_w_delta.rename(columns = {"bottles_purchased" : "avg_bottles_purchased", "purchase_dollars" : "avg_purchase_dollars"}, inplace=True)
fig= px.line(means_w_delta, x="start_epoch",
                    y="avg_purchase_dollars",
                    title='Purchase Dollars Per Line Item  - Jan-2024, Window Size=4 hours, Stride=2 hours')
fig.update_xaxes(
```

```
        rangebreaks=[
            dict(bounds=["sat", "mon"]),
            dict(bounds=[19, 7], pattern="hour"), #hide non-business hours

        ]
    )

fig.add_hline(y= delta_thresh, line_dash="dot",
              annotation_text=f"${round(delta_thresh, 2)} Mean - Yr 2023-Inflation Adjusted + 10%",
              annotation_position="top right")
Image(fig.to_image(format="png", width= 1200, height= 700,scale=2))
```

Out[ ]:




With Delta Encoding, there are much fewer periods reported, as shown by the smoother curve in the chart, resulting in less memory and computation of from data processing and reporting, in addition to meeting the requirement for reporting based on the specified criteria. This can be a fast way to identify increased resource needs at the affected parts of the organization, allowing leaders to respond faster.

## Exponential Smoothing and Forecasting

This section utilizes the Statsmodels package to model Exponential Smoothing using various methods and hyperparameters.

In [ ]:
```
## initialize the streamer package for the target date range
streamer = StreamingAnalytics(in_data    = comparison_df
                             ,start_date = '2023-01-02',
                              end_date   = '2024-12-31'
)
```

**Reporting Intervals**

To run analysis on the data, this assumes that aggregates are calculated on a fixed interval. The streaming aggregates are applied at hourly and daily intervals. Additionally, the function below can be used for daily and minutely intervals. However, this analysis will focus on hourly aggregations.

In [ ]:
```
cols_list = ["txn_datetime","date", "store_number", "vendor_number", "bottles_purchased", "purchase_dollars"]
aggs_dict = {"bottles_purchased": "sum", "purchase_dollars" :"sum"}

def groupby_timeframe( data,by, cols, aggs_dict):
  """
   Function to group data by timeframe
   Params:
      data      - DataFrame of the target
      by        - date columns for aggregation
      columns   - the subset of columns to get from the the supplied data
      aggs_dict - Dictionary of aggegrates for the .agg portion of the .groupby expression
  """
  by = by.lower().strip()
  ord_totals = None

  if  by in ["date", "day", "daily"] :

    date_col = "date"
    ord_totals  = data[cols].groupby([date_col]).agg(aggs_dict)

  if by in ["hr","hour", "hourly", "1hr" ]:

    date_col = "hr"
    cols.append(date_col)

    if  "hr" in data.columns: data.drop(columns="hr")
    data["hr"]    = pd.to_datetime(data.txn_datetime.dt.strftime('%Y%m%d%H'), format='%Y%m%d%H')

    ord_totals  = data[cols].groupby([date_col]).agg(aggs_dict)

  if by in ["min", "minute", "1min", "minutely"] :
    date_col = "min"
    cols.append(date_col)

    if  "min" in data.columns: data.drop(columns="min")

    data["min"]   = pd.to_datetime(data.txn_datetime.dt.strftime('%Y%m%d%H%M') , format='%Y%m%d%H%M')
    ord_totals  = data[cols].groupby(["min"]).agg(aggs_dict)

  ord_totals.reset_index( inplace=True)

  return date_col, ord_totals
```

In [ ]:
```
date_col, order_totals = groupby_timeframe(streamer.pre_df, "hr", cols_list, aggs_dict)
order_totals.head()
```

Out[ ]:

| | hr | bottles_purchased | purchase_dollars |
|---|---|---|---|
| **0** | 2023-01-02 08:00:00 | 43 | 487.05 |
| **1** | 2023-01-02 09:00:00 | 81 | 1323.3 |
| **2** | 2023-01-02 10:00:00 | 116 | 2244.69 |
| **3** | 2023-01-02 11:00:00 | 29 | 767.88 |
| **4** | 2023-01-02 12:00:00 | 133 | 1789.47 |

The next two functions below assign dates in the future to forecast data. This is mainly done to keep the charts date axis in line with expected business hours.

In [ ]:

```
x = order_totals[date_col]

def get_fcast_times(x, fcast_periods, date_col, num_day_hrs = 10):

    raw_fcast_times = pd.Series([(x.loc[len(x)-1] +pd.Timedelta(hours=a)) for a in range(1,fcast_periods+1)])


    ## handle_current_date by removing adding any of the current_day_forevast values
    times_same_day = raw_fcast_times[ (raw_fcast_times.dt.strftime("%Y%m%d")== x.loc[len(x)-1].strftime("%Y%m%d")) & \
                                      (raw_fcast_times.dt.strftime("%H").astype(int) <=17)]

    ##remove handled periods
    if len(times_same_day) > 0 :
      raw_fcast_times  = raw_fcast_times[~(raw_fcast_times.isin(times_same_day))]
      raw_fcast_times.reset_index(drop=True, inplace=True)

    num_fcast_days = len(raw_fcast_times)/num_day_hrs if len(raw_fcast_times)%num_day_hrs ==0 else round(len(raw_fcast_times)//num_day_hrs) + 1
    num_fcast_days = int(num_fcast_days)

    new_dates = [pd.to_datetime(x.loc[len(x)-1].strftime('%Y%m%d'), format= '%Y%m%d') +  pd.Timedelta(days=d)  for d in range(1,num_fcast_days+1)]
    new_raw = raw_fcast_times


    for idx, new_date in enumerate(new_dates):
          start =  idx * num_day_hrs

          if (idx + 1) == len(new_dates) and len(new_raw)%num_day_hrs > 0 :
              end = start + (len(new_raw)%num_day_hrs) -1
              num_day_hrs = (len(new_raw)%num_day_hrs)

          else:
            end   =  start + (num_day_hrs-1)

          new_raw.loc[start:end] = [new_date + pd.Timedelta( hours= 8 + h ) for h in range(0,num_day_hrs)]


    if len(times_same_day) > 0:
      x_out = pd.concat([times_same_day, new_raw], ignore_index=True)
    else:
      x_out = new_raw
    return x_out



fcast_dates = get_fcast_times(x, fcast_periods =100  , date_col = "hr", num_day_hrs = 10)
```

This function offsets hourly timestamps originally set on weekends to Mondays or Tuesdays, while offsetting the result of of the dates so every timestamp is unique.

In [ ]:
```
def remove_weekends(fcast_dates):

    filter = (fcast_dates.dt.day_name().isin(["Saturday", "Sunday"]))
    target_dates= fcast_dates[filter]
    counter = 0

    while len(target_dates) > 0 :

        new_dates  = target_dates + pd.Timedelta(days=2)
        orig_weekdays_offset = len(new_dates.dt.day_name().unique())


        # Move Saturday to Monday if only Saturday
        if orig_weekdays_offset ==1:

          if new_dates.dt.day_name().unique() == 'Saturday':
            target_dates_offset = 2
          else:
            target_dates_offset = 1
        else:
          target_dates_offset = orig_weekdays_offset

        fcast_dates.loc[fcast_dates >= new_dates.min() ] = fcast_dates.loc[fcast_dates >= new_dates.min() ] + pd.Timedelta(days=orig_weekdays_offset)
        fcast_dates.loc[target_dates.index] = fcast_dates.loc[target_dates.index] + pd.Timedelta(days=target_dates_offset)
        target_dates= fcast_dates[fcast_dates.dt.day_name().isin(["Saturday", "Sunday"])]

        counter+=1
        if counter >100:
            raise ValueError("Too many weekend date offset iterations. Consider Reducing the length of the forecast")
            break


    return fcast_dates

fcast_dates = remove_weekends(fcast_dates)
```

In [ ]:
```
print(f"Highest Number of rows per datetime (uniqueness check): {fcast_dates.value_counts().max()}")
```

```
Highest Number of rows per datetime (uniqueness check): 1
```

In [ ]:
```
# Check the dates
fcast_dates.head(10)
```

Out[ ]:

| | 0 |
|---|---|
| **0** | 2024-08-01 08:00:00 |
| **1** | 2024-08-01 09:00:00 |
| **2** | 2024-08-01 10:00:00 |
| **3** | 2024-08-01 11:00:00 |
| **4** | 2024-08-01 12:00:00 |
| **5** | 2024-08-01 13:00:00 |
| **6** | 2024-08-01 14:00:00 |
| **7** | 2024-08-01 15:00:00 |
| **8** | 2024-08-01 16:00:00 |
| **9** | 2024-08-01 17:00:00 |

**dtype:** datetime64[ns]

### Simple Exponential Smoothing

This section uses the Simple Exponential Smoothing function from the StatsModels package. This is a first order implementation that takes $Alpha$, the Decay Constant as a parameter. The idea behind exponential smoothing is that past values become less important the further back are from present time. The smaller the value of $Alpha$ or the $smoothing - level$, the larger the smoothing effect. A hyperparameter search will be used to determine the best outcome for this data. The idea is to use the last two week's data, or last 100 hours to forecast data for the next 1 to 5 days. In this case, both bottles_purchased and purchase_dollars can be used, but this method can be used for any numeric data . Since we are assuming only the use of the last 100 data points to 1000 data points at a time, this also works well for streaming analytics because of the relatively low memory and computational requirements. To evaluate the models, the Root Mean Squared Error will be used. The StasModels classes in this implementation provide the Sum of Squared Errors value. However, a simple custom function will be used here to calculate the Root Mean Squared Error since that outcome is more aligned with the original units of the data, and therefore easier to interpret.

See https://www.statsmodels.org/dev/examples/notebooks/generated/exponential_smoothing.html for reference.

In [ ]:
```
# This returns the Root Mean Squared Error of the Smoothing Algorithm
def get_rmse(sse, n ):
  return (sse/n)**0.5
```

### Initial Hyper-Parameter Search

Initially, all possible hyperparameters will be combined. Depending on the efficiency of the process, random selection might be used to sample the combinations, otherwise all combinations of parameters will be used.

In [ ]:
```
def get_simpleexp_sm(  data,
                       smoothing_params = [],
                       fcast_periods=5,
                       optimized=False ):

    fits   = [ SimpleExpSmoothing(data,
                                  initialization_method=p[0]) \
               .fit( smoothing_level= p[1], optimized= optimized) for p in smoothing_params
             ]

    fcasts   = [fit.forecast(fcast_periods) for fit in fits]

    fit_rmse = [get_rmse(fit.sse, len(fit.fittedvalues)) for fit in fits ]

    return fits, fcasts , fit_rmse
```

**Plots of Best and Worst Outcomes with Simple/First Order Exponential Smoothing**

In [ ]:

```
def plot_smoothing_lines(x, y,
                         fitted_obj,
                         forecast_obj,
                         title,
                         show_forecast=True,
                         as_image=True,
                         image_width  = 1800,
                         image_format = "png",
                         image_scale  = 2.0):
    # Create traces
    fig = go.Figure(  layout=go.Layout(
                                    title=go.layout.Title(text=title)
                                      )
                  )

    alpha = fitted_obj.params["smoothing_level"]
    beta  = fitted_obj.params["smoothing_trend"]

    smoothed_name =  f'alpha={alpha} , \n beta={beta}'



    fig.add_trace(go.Scatter(x=x, y=y,
                        mode='lines+markers',
                        name='Original Series'))

    fig.add_trace(go.Scatter(x=x, y=fitted_obj.fittedvalues,
                        mode='lines',
                        name=f'{smoothed_name}'))

    if show_forecast:
        fcast_dates = get_fcast_times(x, len(forecast_obj), "hr",10)
        fcast_dates = remove_weekends(fcast_dates)
        fig.add_trace(go.Scatter(x=fcast_dates,
                                 y=forecast_obj,
                                 mode='lines+markers',
                                 name=f'Forecast'))

    fig.update_xaxes(
        rangebreaks=[
            dict(bounds=["sat", "mon"]),
            dict(bounds=[23, 7], pattern="hour"), #hide hours

        ]
    )

    fig.update_layout(legend=dict(
    orientation="h",
    yanchor="bottom",
    y=1.02,
    xanchor="right",
    x=1
    ))

    if as_image:
      img = Image(fig.to_image(format=image_format, width = image_width, scale = image_scale))
      return img
    else:
      fig.show()
```

This function will display the best and worst performing plots based on rmse for the selected Criteria.

In [ ]:
```
from IPython.display import display

def show_best_worst(x, y, fits, fcasts, smoothing_params, rmse, title_prefix='', best_only = False):

    print("Min RMSE: " ,np.min(rmse))
    print("Max RMSE: ", np.max(rmse))

    best_idx = np.argmin(rmse)
    worst_idx = np.argmax(rmse)

    print(f"Best Params :  {smoothing_params[best_idx]}, idx={best_idx}")
    print(f"Worst Params: {smoothing_params[worst_idx]}, idx={worst_idx}")

    display(plot_smoothing_lines(x, y, fits[best_idx], fcasts[best_idx], title   = f"{title_prefix} Best- RMSE"))
    if not best_only:
      display(plot_smoothing_lines(x, y, fits[worst_idx],fcasts[worst_idx], title = f"{title_prefix} Worst- RMSE"))

    return best_idx, worst_idx
```

For simplicity, this will assume that every business day is always 10 hours, excluding weekends. The data is further limited so analysis can be done for the last 100 hours, which equates to 2-week intervals.

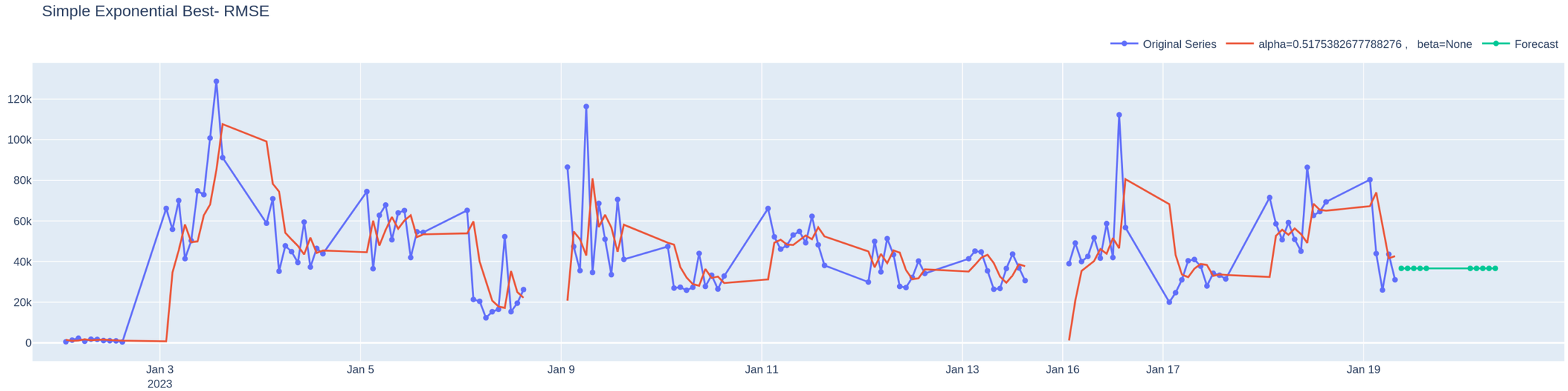
In [ ]:
```
import itertools
np.random.seed(265)
smoothing_alphas = np.random.uniform(0.1,0.9, 20)
smoothing_alphas
init_methods     = ["estimated", "heuristic"]
params           = [init_methods, smoothing_alphas]
smoothing_params = list(itertools.product(*params))
print(f"Number of Combinations {len(list(smoothing_params))}" )
```

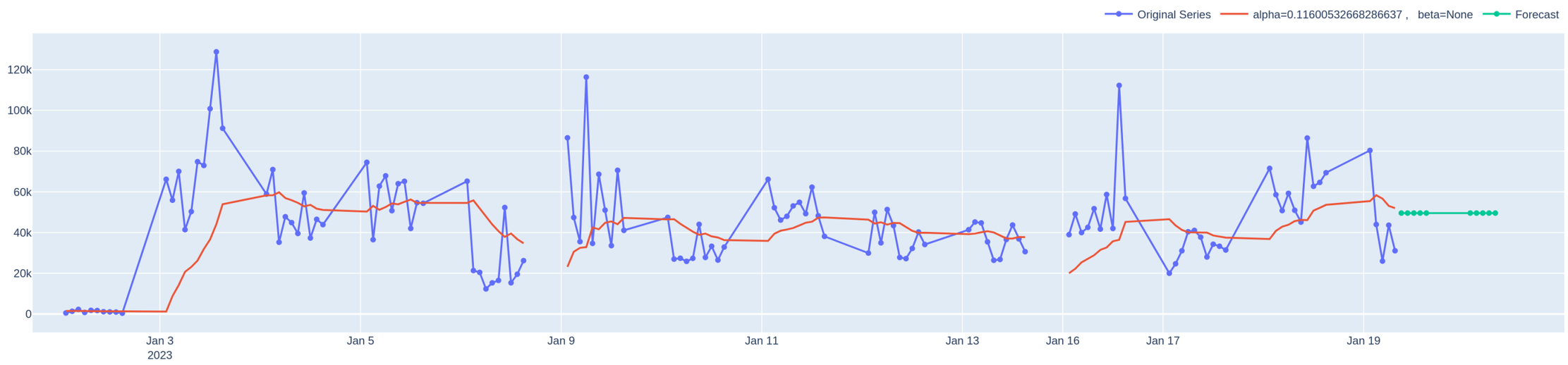
```
Number of Combinations 40
```

In [ ]:
```
limit = 150
y                   = order_totals.purchase_dollars.loc[0:limit-1].tolist()
x                   = order_totals.hr.loc[0:limit-1]

fits,fcasts, fit_rmse = get_simpleexp_sm( data = y,
                                          smoothing_params= smoothing_params,
                                          fcast_periods=10,
                                          optimized    =False


                )
```

In [ ]:
```
title_prefix = "Simple Exponential"

show_best_worst(x,y,  fits, fcasts,smoothing_params,fit_rmse,  "Simple Exponential" )
```

```
Min RMSE:  19326.66708449149
Max RMSE:  22623.63233547938
Best Params :  ('estimated', 0.5175382677788276), idx=0
Worst Params: ('estimated', 0.11600532668286637), idx=3
```




Out[ ]: (0, 3)

It is very obvious from the charts that as expected, the lower RMSE result is a better fit in this case. However, also as exepced, the forecast is not really useful. It is just a flat line with no variation for the 10 forcasted hours.

The next section will repeat the process, however, this time a trend or $beta$ parameter will be added such that the forecast will have a slope.

**Second Order Exponential Smoothing/ Holt's Method**

It is obvious from the charts that as expected, the lower RMSE result is a better fit in this case. However, also as expected, the forecast is not useful. It is just a flat line with no variation for the 10 forecast periods.

The next section will repeat the process, however, this time a trend or $beta$ parameter will be added such that the forecast will have a slope.

In [ ]:
```
np.random.seed(4465897)
smoothing_betas = np.random.uniform(0.1, 0.9, 20)
init_methods     = ["estimated", "heuristic"]
exponential      = [True, False]
optimized        = [False]
params           = [exponential, init_methods, smoothing_alphas, smoothing_betas, optimized]
smoothing_params = list(itertools.product(*params))
print(f"Number of Combinations {len(list(smoothing_params))}" )

smoothing_params[0]
```

Out[ ]:
```
Number of Combinations 1600
(True, 'estimated', 0.5175382677788276, 0.2666279779751084, False)
```

In [ ]:
```
def get_doubleexp_sm( data,
                      smoothing_params=[],
                      fcast_periods=10,

                     ):

    fits   = [ Holt(data,
                    exponential   =p[0],
                    initialization_method=p[1]) \
               .fit( smoothing_level = p[2],
                     smoothing_trend = p[3] ,
                     optimized       = p[4]) for p in smoothing_params
             ]

    fcasts = [fit.forecast(fcast_periods) for fit in fits]

    fit_rmse = [ get_rmse(fit.sse, len(fit.fittedvalues)) for fit in fits]

    return fits, fcasts, fit_rmse


holt_fits,holt_fcasts, holt_fit_rmse = get_doubleexp_sm( data = y,
                                       smoothing_params = smoothing_params ,
                                       fcast_periods=10
                                     )
```

In [ ]:
```
show_best_worst(x,y,  holt_fits, holt_fcasts,smoothing_params,holt_fit_rmse,  "Second Order/Holt's Exponential" )
```

```
Min RMSE:  20710.63015349396
Max RMSE:  471450418070.10004
Best Params :  (False, 'estimated', 0.4779329842682727, 0.25286049126848315, False), idx=1055
Worst Params: (True, 'estimated', 0.11600532668286637, 0.4082413322633396, False), idx=63
```





Out[ ]:
```
(1055, 63)
```

The worst RMSE case is degenerative, further illustrating that RMSE is a good evaluation metric in this case. Next, the case with the RMSE in the middle of the series will be used.

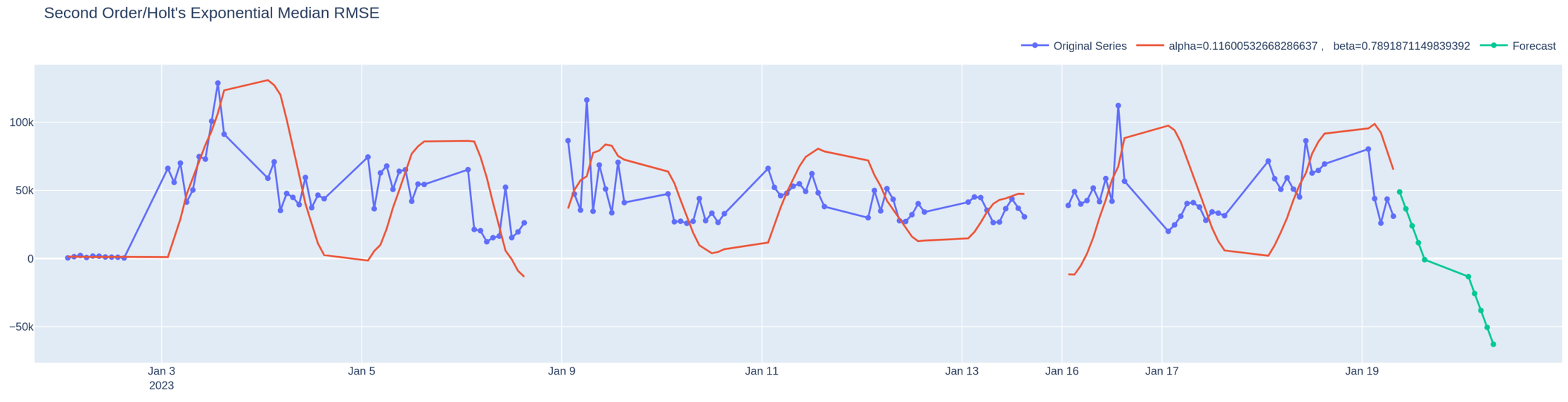
In [ ]:
```
print("Values at midpoints or sorted array: ")
print(f"First Value sorted: {sorted(holt_fit_rmse)[int(len(holt_fit_rmse)/2) - 1]}")
print(f"Second Value sorted: {sorted(holt_fit_rmse)[int(len(holt_fit_rmse)/2)]}")
median_idx = np.argwhere(holt_fit_rmse == sorted(holt_fit_rmse)[int(len(holt_fit_rmse)/2) - 1])[0][0]
median_idx
```

```
Values at midpoints or sorted array:
First Value sorted: 32141.780498406166
Second Value sorted: 2330202.935216178
870
```

Out[ ]:

In [ ]:
```
plot_smoothing_lines(x, y, holt_fits[median_idx], holt_fcasts[median_idx], title = "Second Order/Holt's Exponential Median RMSE")
```

Out[ ]:




It is obvious from the smoothing that in January 2023, there is more buying activity at the beginning of the week, followed by dips in the middle of the week, followed by an trend and another dip towards the end of the week. We will examine whether this trend persists throughout the analysis period later. However, for forecasting, in this case, we will conclude that the Best RMSE value is the most reasonable choice for providing a forecast.

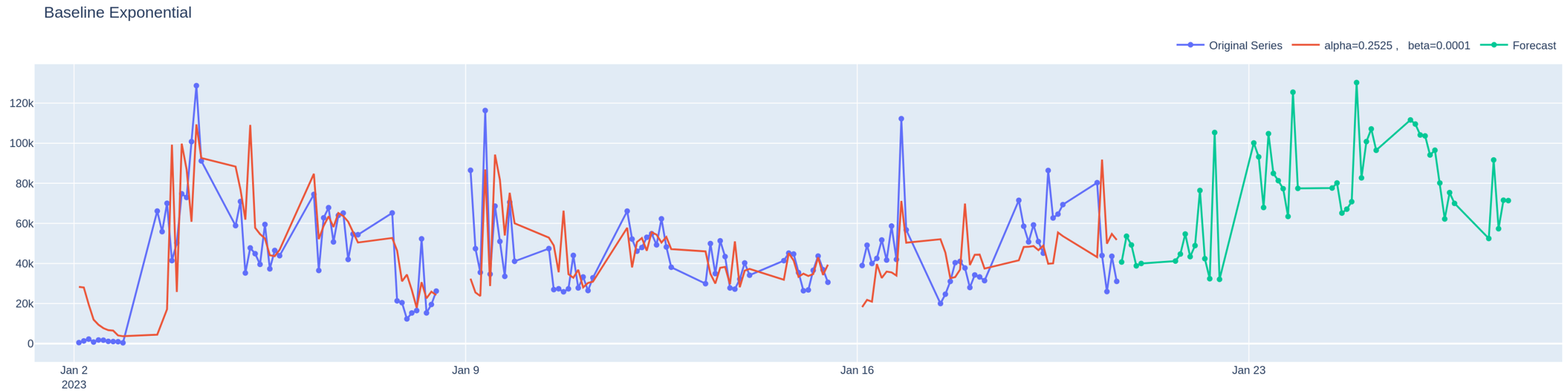
**Baseline Holt Winters**

The baseline assumes that the trend and seasonality are not consistent, thereby making use of the multiplicative instead of the additive method, which generally works better for consistent trends. Adjustments will be made in further analysis to determine if better fits can be attained using longer series and different parameters, including the use of a damped trend parameter. The damped trend has the effect of damping the forecast trajectory, which in general, provides more accurate forecasts.

In [ ]:
```
np.random.seed(89074)
baseline_expo_fit = ExponentialSmoothing(y, trend="mul", seasonal="mul", seasonal_periods=50).fit()
baseline_expo_fcast = baseline_expo_fit.forecast(50)

print(f"Baseline RMSE: {get_rmse(baseline_expo_fit.sse, len(baseline_expo_fit.fittedvalues))}")

plot_smoothing_lines(x, y, baseline_expo_fit, baseline_expo_fcast, title = "Baseline Exponential")
```

```
Baseline RMSE: 19951.08534732745
```

Out[ ]:




In [ ]:

The first and second order forecasts have thus far not provided any useful insight, although useful in understanding the weekly buying patterns. The Holt-Winters method has now yielded the most plausible forecast. From this point on, Exponential Smoothing Analysis , will be the main focus, with an emphasis on Third Order/Holt-Winters Method an forecasts.

**Hyper-Parameter Search**

In addition to the other parameters explained, the box_cox transformation converts the input data to the best bit that resembles a normal distribution. While not necessary here, it could yield better results. This implementation of Holt-Winter's algorithms attempts to find the best alpha, beta and phi (the damping parameter) based on straightforward parameters, without the need to explicitly specify the numerical values.

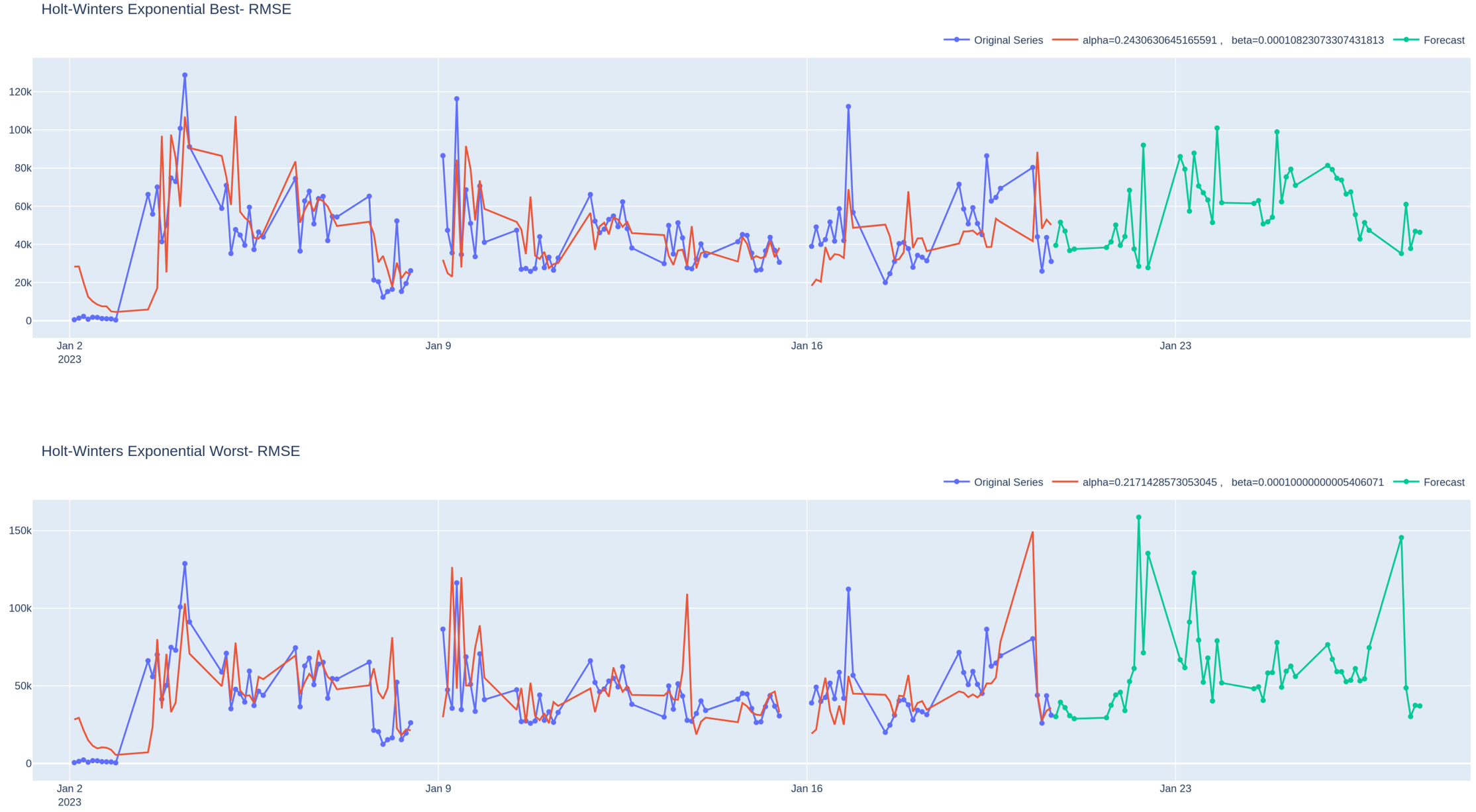
In [ ]:
```
seed = 729800
np.random.seed(seed)

init_methods          = ["estimated", "heuristic"]
trend                 = [ "add", "mul"]
seasonal              = [ "mul"]
damped_trend          = [True, False]
box_cox               = [True, False]
params           = [
                    trend,
                    seasonal,
                    damped_trend,
                    init_methods,
                    box_cox ]
smoothing_params = list(itertools.product(*params))
print(f"Number of Combinations {len(list(smoothing_params))}" )
```

```
Number of Combinations 16
```

In [ ]:
```
np.random.seed(seed)

def get_holt_winters_sm( data,
                         smoothing_params=[],
```

```
                        fcast_periods   =10,
                        seasonal_periods = 50

                    ):

    fits   = [ ExponentialSmoothing(data,
                                    trend       =p[0],
                                    seasonal    =p[1],
                                    damped_trend=p[2],
                                    seasonal_periods= seasonal_periods,
                                    initialization_method = p[3],
                                    use_boxcox    =p[4]
                                    ).fit()
                for p in smoothing_params
             ]

    fcasts = [fit.forecast(fcast_periods) for fit in fits]

    fit_rmse = [ get_rmse(fit.sse, len(fit.fittedvalues)) for fit in fits]

    return fits, fcasts, fit_rmse


hw_fits,hw_fcasts, hw_fit_rmse = get_holt_winters_sm( data = y,
                                      smoothing_params = smoothing_params ,
                                      fcast_periods   =50,
                                      seasonal_periods=50

                                    )


best_idx, worst_idx =show_best_worst(x,y,  hw_fits, hw_fcasts,smoothing_params,hw_fit_rmse,  "Holt-Winters Exponential" )
```

```
Min RMSE:  19722.770645853943
Max RMSE:  24949.543050267246
Best Params :  ('add', 'mul', True, 'heuristic', False), idx=3
Worst Params: ('add', 'mul', True, 'estimated', False), idx=1
```




In [ ]:

The two charts above raise the question of which fit and forecast is better. The next section will compare known data to the forecasts.

In [ ]:
```
## A custom SSE function to compare to compare forecast data to real
def custom_sse( yhat, y ):
  return np.sum(np.power(np.subtract(yhat, y), 2))
```

In [ ]:
```
order_totals.head()
```

Out[ ]:

| | hr | bottles_purchased | purchase_dollars |
|---|---|---|---|
| **0** | 2023-01-02 08:00:00 | 43 | 487.05 |
| **1** | 2023-01-02 09:00:00 | 81 | 1323.3 |
| **2** | 2023-01-02 10:00:00 | 116 | 2244.69 |
| **3** | 2023-01-02 11:00:00 | 29 | 767.88 |
| **4** | 2023-01-02 12:00:00 | 133 | 1789.47 |

From the Printout Above the indices in question are 1 and 12. The next section will match up actual hourly data timestamps with forecasts instead of the artificial timestamps used previously. This will ensure that the forecast data points can match exactly with data points for the comparisons.

In [ ]:
```
np.random.seed(seed)
def forecast_min_rmse(x, next_series, fcasts, data_len, idxs, date_col= "hr", numeric_col="purchase_dollars"):
    "Function that saves values only when the new value is less than any other"
    output   = []
    last_min = np.inf

    for idx in idxs:
        sse = custom_sse(fcasts[idx], next_series.tolist())
        if sse < last_min :
          output = []
          output.append({idx: get_rmse(sse, data_len)})
          last_min = sse
    return output



idxs = [best_idx,worst_idx]
data_len = hw_fcasts[1].size
last_ts = x.loc[len(x)-1]
next_series = order_totals.loc[order_totals[date_col] > last_ts, "purchase_dollars"][0: data_len ]
next_x      = order_totals.loc[order_totals[date_col] > last_ts, "hr"][0: data_len ]
forecast_min_rmse(x, next_series, hw_fcasts, data_len, idxs)
```

Out[ ]:
```
[{3: 26230.98434518136}]
```

In [ ]:
```
##Running for all forecasts
idxs = [r for r in range(0,len(hw_fcasts))]
best = forecast_min_rmse(x, next_series, hw_fcasts, data_len, idxs)
best
```

Out[ ]:
```
[{0: 22805.09237678903}]
```

In this case the best forecast based on the RMSE of the input data and the smoothed series, resulted in a worse outcome when compared to other forecasts. Plotting will give more insight.

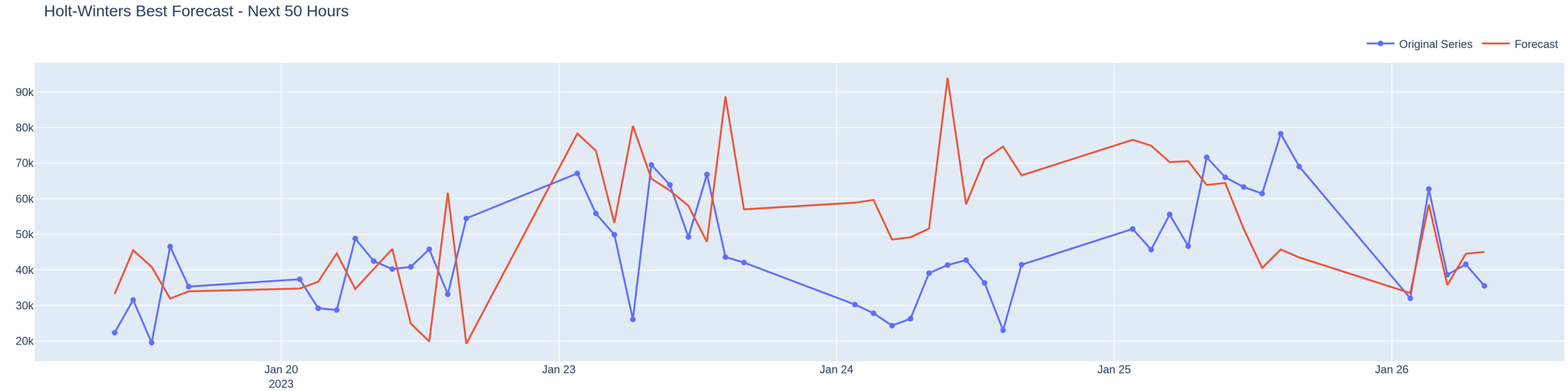
In [ ]:
```
def plot_forecast_lines(x, y, forecast_objs, title, as_image = True, image_format="png", image_width=1800, image_scale=2.0):
    # Create traces
    fig = go.Figure(  layout=go.Layout(
                                    title=go.layout.Title(text=title)
                                      )
                    )
```

```
    fig.add_trace(go.Scatter(x=x, y=y,
                             mode='lines+markers',
                             name='Original Series'))

    for forecast_obj in forecast_objs:
        fig.add_trace(go.Scatter(x=x, y=forecast_obj,
                                 mode='lines',
                                 name='Forecast'))



    fig.update_xaxes(
        rangebreaks=[
            dict(bounds=["sat", "mon"]),
            dict(bounds=[22, 7], pattern="hour"), #hide hours

        ]
    )

    fig.update_layout(legend=dict(
    orientation="h",
    yanchor="bottom",
    y=1.02,
    xanchor="right",
    x=1
    ))

    if as_image:
      img = Image(fig.to_image(format=image_format, width = image_width, scale = image_scale))
      return img
    else:
        fig.show()
```

In [ ]:

```
np.random.seed(seed)
best_index =  list(best[0].keys())[0]
plot_forecast_lines(x = next_x, y=next_series, forecast_objs= [hw_fcasts[best_index]], title = f"Holt-Winters Best Forecast - Next {data_len} Hours")
```

Out[ ]:




**Experimentation with Input Data Lengths**

The comparison chart shows that the forecast could be a plausible indicator of future peaks and troughs. In the next section, analysis will be conducted to determine if changing the lengths of the input and forecast data will result in a better forecast.

In [ ]:
```
order_totals.head()
```

Out[ ]:

| | hr | bottles_purchased | purchase_dollars |
|---|---|---|---|
| **0** | 2023-01-02 08:00:00 | 43 | 487.05 |
| **1** | 2023-01-02 09:00:00 | 81 | 1323.3 |
| **2** | 2023-01-02 10:00:00 | 116 | 2244.69 |
| **3** | 2023-01-02 11:00:00 | 29 | 767.88 |
| **4** | 2023-01-02 12:00:00 | 133 | 1789.47 |

In [ ]:
```
order_totals.tail()
```

Out[ ]:

| | hr | bottles_purchased | purchase_dollars |
|---|---|---|---|
| **4616** | 2024-07-31 13:00:00 | 3020 | 51282.53 |
| **4617** | 2024-07-31 14:00:00 | 2205 | 37813.83 |
| **4618** | 2024-07-31 15:00:00 | 2130 | 36436.34 |
| **4619** | 2024-07-31 16:00:00 | 4850 | 66951.05 |
| **4620** | 2024-07-31 17:00:00 | 3440 | 67319.51 |

This provides a more flexible dataset to experiment with. In the next section, the length of the input data will be lengthened while using the same 50 -hour forecast, using the parameters that gave the best fit and forecast thus far.

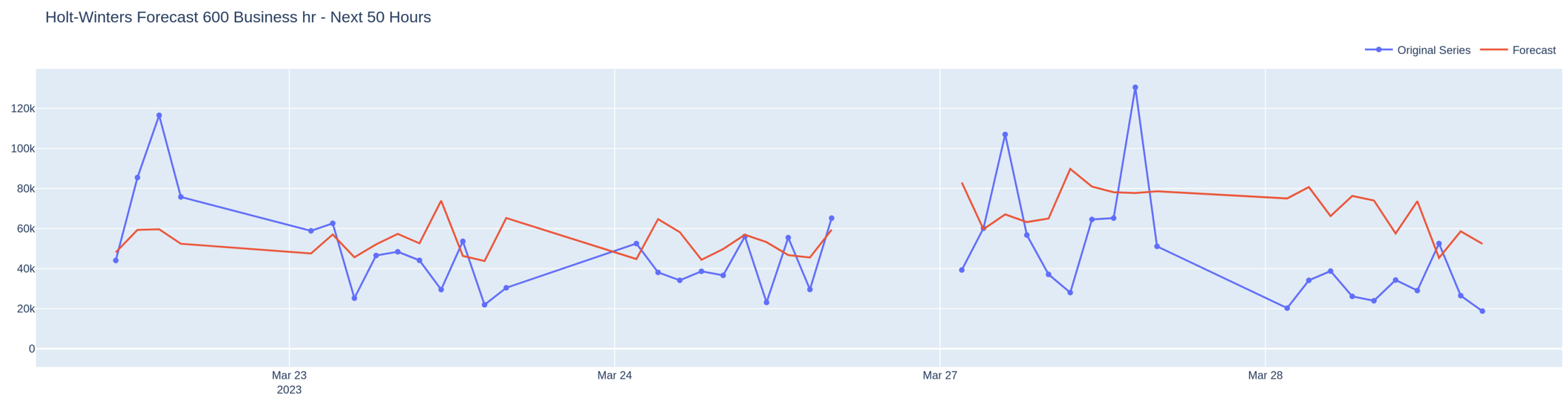
In [ ]:
```
##Roughly 3 month limit
np.random.seed(seed)
limit = 3 * 4 * 50
y                   = order_totals.purchase_dollars.loc[0:limit-1].tolist()
x                   = order_totals.hr.loc[0:limit-1]

forecast_eval_smoothing_params =  [smoothing_params[best_index]]
print(forecast_eval_smoothing_params)
hw_fits,hw_fcasts_600, hw_fit_rmse = get_holt_winters_sm( data = y,
                                                          smoothing_params = forecast_eval_smoothing_params ,
                                                          fcast_periods   =50,
                                                          seasonal_periods=50

                                                        )
```

```
[('add', 'mul', True, 'estimated', True)]
```

There is only one forecast series here so zero will be used as the index.

In [ ]:
```
idxs = [0]
data_len = hw_fcasts_600[0].size
last_ts = x.loc[len(x)-1]
next_series = order_totals.loc[order_totals[date_col] > last_ts, "purchase_dollars"][0: data_len ]
next_x      = order_totals.loc[order_totals[date_col] > last_ts, "hr"][0: data_len ]

forecast_min_rmse(x, next_series, hw_fcasts_600, data_len, idxs)

plot_forecast_lines(x = next_x, y=next_series, forecast_objs = [hw_fcasts_600[0]], title = f"Holt-Winters Forecast {limit} Business {date_col} - Next {data_len} Hours")
```

Out[ ]:



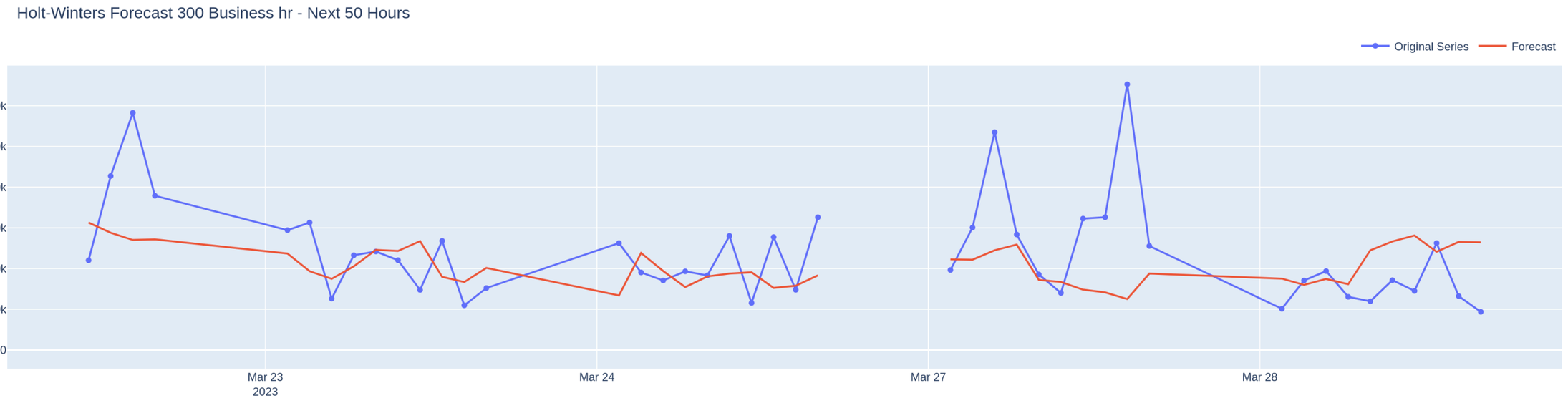

This resulted in higher RMSE. The next implementation will try the same forecast period with less input data.

**300 hours**

In [ ]:
```
y_new =order_totals.purchase_dollars.loc[0:599]
x_new =order_totals.hr.loc[0:599]
```

In [ ]:
```
np.random.seed(seed)
limit = 300
y = list(y_new[-300:].reset_index(drop=True))
x = x_new[-300:].reset_index(drop=True)
hw_fits,hw_fcasts_300, hw_fit_rmse = get_holt_winters_sm( data = y,
                                                  smoothing_params = forecast_eval_smoothing_params ,
                                                  fcast_periods   =50,
                                                  seasonal_periods=50

                                                 )


idxs = [0]
data_len = hw_fcasts_300[0].size
last_ts = x.loc[len(x)-1]
next_series = order_totals.loc[order_totals[date_col] > last_ts, "purchase_dollars"][0: data_len ]
next_x      = order_totals.loc[order_totals[date_col] > last_ts, "hr"][0: data_len ]

plot_forecast_lines(x = next_x, y=next_series, forecast_objs = [hw_fcasts_300[0]], title = f"Holt-Winters Forecast {limit} Business {date_col} - Next {data_len} Hours")
```

Out[ ]:



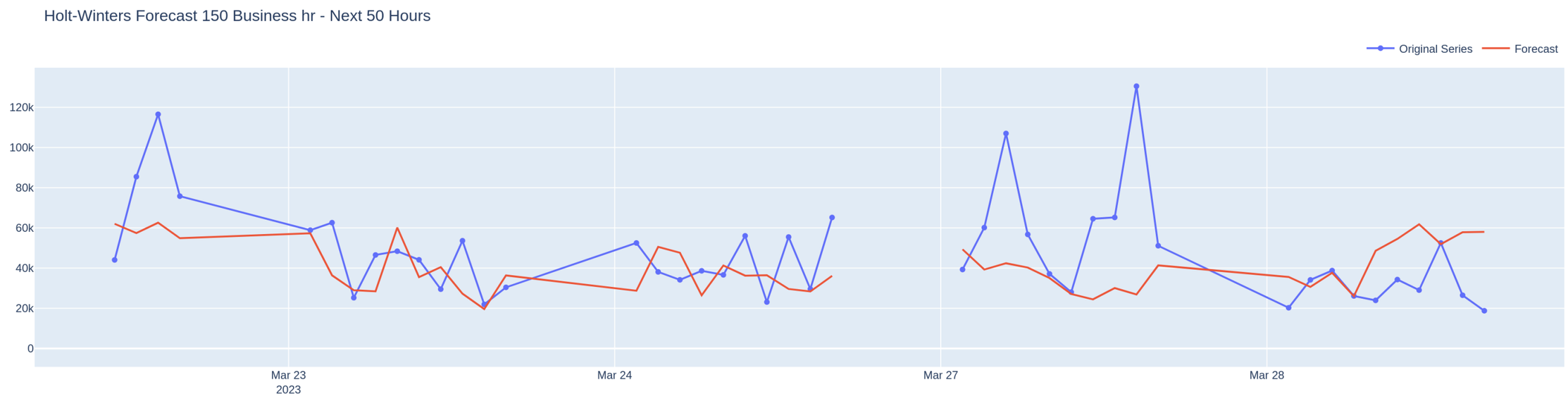

This resulted is a better RMSE Value for the Forecast. Next is the forecast using the original 150 Hours.

**150 hours:**

In [ ]:
```
np.random.seed(seed)

limit = 150
y = list(y_new[-150:].reset_index(drop=True))
x = x_new[-150:].reset_index(drop=True)

hw_fits,hw_fcasts_150, hw_fit_rmse = get_holt_winters_sm( data = y,
```

```
                                                smoothing_params = forecast_eval_smoothing_params ,
                                                fcast_periods   =50,
                                                seasonal_periods=50

                                               )

idxs = [0]
data_len = hw_fcasts_150[0].size
last_ts = x.loc[len(x)-1]
next_series = order_totals.loc[order_totals[date_col] > last_ts, "purchase_dollars"][0: data_len ]
next_x      = order_totals.loc[order_totals[date_col] > last_ts, "hr"][0: data_len ]

plot_forecast_lines(x = next_x, y=next_series, forecast_objs = [hw_fcasts_150[0]], title = f"Holt-Winters Forecast {limit} Business {date_col} - Next {data_len} Hours")
```

Out[ ]:



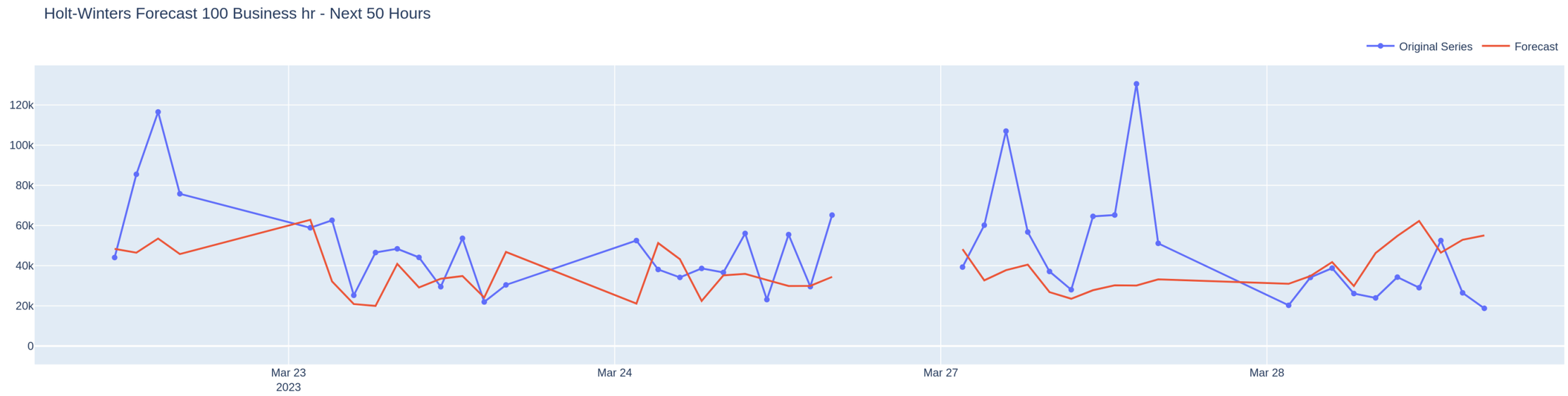

**100 Hours:**

In [ ]:

```
np.random.seed(seed)

limit = 100
y = list(y_new[-100:].reset_index(drop=True))
x = x_new[-100:].reset_index(drop=True)

hw_params = forecast_eval_smoothing_params


## Handle issue with Multiplicative Seasonal and Trend for shortest allowable period, when Box-cox == True

if ('mul', 'mul', True, 'estimated', True) == forecast_eval_smoothing_params[0]:
  ##simply ignore the box-cox transformation
  hw_params = [('mul', 'mul', True, 'estimated', False)]

hw_fits,hw_fcasts_100, hw_fit_rmse = get_holt_winters_sm( data = y,
                                                smoothing_params = hw_params,
                                                fcast_periods   =50,
                                                seasonal_periods=50
```

```
                                        )



idxs = [0]
data_len = hw_fcasts_100[0].size
last_ts = x.loc[len(x)-1]
next_series = order_totals.loc[order_totals[date_col] > last_ts, "purchase_dollars"][0: data_len ]
next_x      = order_totals.loc[order_totals[date_col] > last_ts, "hr"][0: data_len ]

plot_forecast_lines(x = next_x, y=next_series, forecast_objs = [hw_fcasts_100[0]], title = f"Holt-Winters Forecast {limit} Business {date_col} - Next {data_len} Hours")
```

Out[ ]:




**A full year and a Half +**

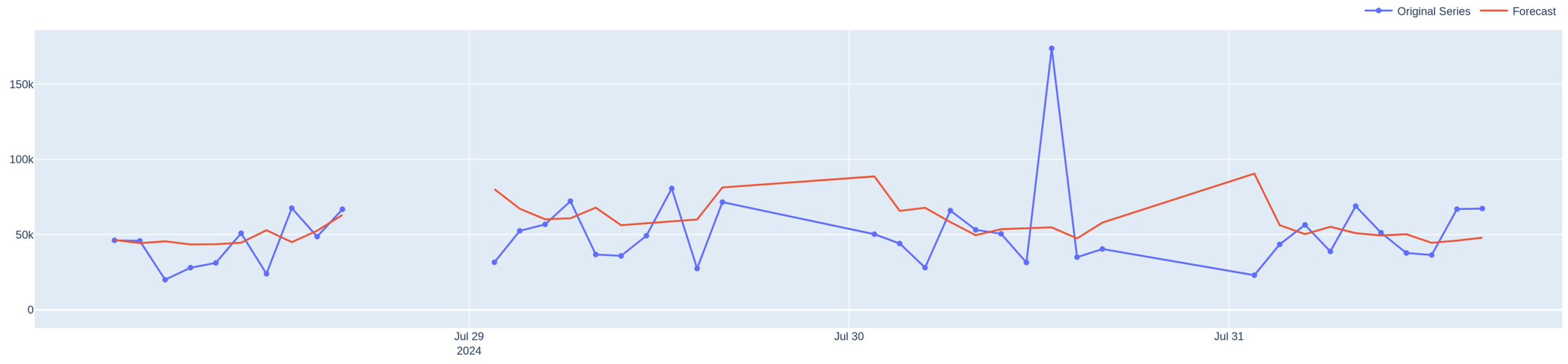
Finally, the next section does analysis on the full dataset, leaving 50 hours for forecast analysis in the last week of July 2024.

In [ ]:
```
series_len = len(order_totals) - 51
y =order_totals.purchase_dollars.loc[0: series_len]
x =order_totals.hr.loc[0:series_len]
```

In [ ]:
```
np.random.seed(seed)
limit = len(x)
y = list(y.reset_index(drop=True))
x = x.reset_index(drop=True)


hw_fits,hw_fcasts_all, hw_fit_rmse = get_holt_winters_sm( data = y,
                                                          smoothing_params = forecast_eval_smoothing_params ,
                                                          fcast_periods   =50,
                                                          seasonal_periods=50

                                                        )
```

```
idxs = [0]
data_len = hw_fcasts_all[0].size
last_ts = x.loc[len(x)-1]
next_series = order_totals.loc[order_totals[date_col] > last_ts, "purchase_dollars"][0: data_len ]
next_x      = order_totals.loc[order_totals[date_col] > last_ts, "hr"][0: data_len ]

plot_forecast_lines(x = next_x, y=next_series, forecast_objs = [hw_fcasts_all[0]], title = f"Holt-Winters Forecast {limit} Business {date_col} - Next {data_len} Hours")
```

Out[ ]:




**Exponential Smoothing Conclusions**

The general observation is that the lower the input data length, the closer the forecast is to the real data. There is a requirement that the input data must be at least twice the length of the seasonal parameter. On the other hand, lengthening the input data seems to create a more consistent, smoother curve that seems to generally be an average that can be used to indicate a crossing point to which the data is expected to return and cross after reaching a peak or trough. Combining forecasts generated using input data of different lengths on the same chart might also provide clearer insight. Keeping in mind that the hourly data was simulated using real data aggregated at daily intervals from the source, the hourly forecast could be cautiously used as an indicator of hourly peak and trough patterns for the forecast period, keeping the statistical error in consideration. Increasing the reporting units to 4, or 6 hours might be the next step in the analysis.

# Market Basket Analysis:

The next section attempts to gather insight on frequent items sets; these are items generally purchased together in orders from 2023 to 2024. This analysis is not in the context of streaming data. The full historical subset will be used.

In [ ]:
```
!pip install mlxtend==0.23.1
```

```
Requirement already satisfied: mlxtend==0.23.1 in /usr/local/lib/python3.10/dist-packages (0.23.1)
Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.10/dist-packages (from mlxtend==0.23.1) (1.13.1)
Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend==0.23.1) (1.26.4)
Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend==0.23.1) (2.1.4)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend==0.23.1) (1.3.2)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from mlxtend==0.23.1) (3.7.1)
Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend==0.23.1) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend==0.23.1) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend==0.23.1) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend==0.23.1) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend==0.23.1) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend==0.23.1) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend==0.23.1) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend==0.23.1) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend==0.23.1) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend==0.23.1) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend==0.23.1) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.2->mlxtend==0.23.1) (3.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend==0.23.1) (1.16.0)
```

In [ ]:
```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules, fpgrowth

warnings.simplefilter("ignore", category=DeprecationWarning)
```

To run all the analysis, the Hyvee Stores dataframe from earlier will be used.

In [ ]:
```
df_hyvee.columns
```

Out[ ]:
```
Index(['invoice_and_item_number', 'date', 'store_number', 'store_name', 'city',
       'zip_code', 'county', 'category', 'category_name', 'vendor_number',
       'vendor_name', 'item_number', 'item_description', 'pack',
       'bottle_volume_ml', 'state_bottle_cost', 'state_bottle_retail',
       'bottles_sold', 'sale_dollars', 'volume_sold_liters',
       'volume_sold_gallons', 'purchase_ts'],
      dtype='object')
```

**Assigning Porduct Names**

The Item Description, volume of bottle in ml and size of pack will be used to create unique product names.

In [ ]:
```
product_names = df_hyvee[["item_number", "item_description", "pack", "bottle_volume_ml"]].groupby(["item_number", "item_description", "pack", "bottle_volume_ml"]).count()
product_names.reset_index( inplace =True)
product_names["product_name"] = product_names.item_description.astype(str) + ' ' +  product_names.bottle_volume_ml.astype(str) + 'ml (' + product_names.pack.astype(str) +  ' pack)'

product_names = product_names[product_names.item_number.isin(comparison_df.item_number)]
product_names.index = product_names.item_number
product_names.drop(columns = ["item_number", "item_description",        "pack", "bottle_volume_ml"], inplace=True)

product_names.head()
```

Out[ ]:

| item_number | product_name |
|---|---|
| **100026** | ABSOLUT W/FEVER TREE GINGER BEER 750ml (6 pack) |
| **10006** | SCORESBY RARE SCOTCH 750ml (12 pack) |
| **10008** | SCORESBY RARE SCOTCH 1750ml (6 pack) |
| **10009** | SCORESBY RARE SCOTCH 1000ml (12 pack) |
| **100104** | HERRADURA ULTRA ANEJO 750ml (6 pack) |

Examination of the data showed that item_number and descriptions are not entirely unique. Duplicates are dropped so that every product/item_number combination is unique.

In [ ]:
```
##filter to relevant products only
product_names.drop_duplicates(inplace=True)
product_names.product_name.value_counts().sort_values(ascending=False).head(5)
```

Out[ ]:

| product_name | count |
|---|---|
| **ABSOLUT W/FEVER TREE GINGER BEER 750ml (6 pack)** | 1 |
| **DIPLOMATICO RESERVA EXCLUSIVA W/ 2 ROCKS GLASSES 750ml (6 pack)** | 1 |
| **SCORESBY RARE SCOTCH 1000ml (12 pack)** | 1 |
| **HERRADURA ULTRA ANEJO 750ml (6 pack)** | 1 |
| **JACK DANIELS SINGLE BARREL W/SNIFTER GLASS 750ml (6 pack)** | 1 |

**dtype:** int64

In [ ]:
```
print( "Number of Products :", len(product_names))
```

```
Number of Products : 4702
```

To efficiently process the data, the order data will need to be converted into a sparse dataframe. Because the original data did not have unique order numbers, this implementation uses the po_number columns added in the streaming data section. The mlextend package Transcoder class will be used to preprocess the data for tranformation into sparse dataframe.

In [ ]:
```
basket_list = comparison_df.loc[:,['po_number', 'item_number']].groupby('po_number')['item_number'].apply(list).tolist()
te = TransactionEncoder()
oht_ary = te.fit(basket_list).transform(basket_list, sparse=True)
sparse_basket_df = pd.DataFrame.sparse.from_spmatrix(oht_ary, columns=te.columns_)

sparse_basket_df.columns = [str(i) for i in sparse_basket_df.columns]
sparse_basket_df.head(10)
```

Out[ ]:

| | 100026 | 10006 | 10008 | 10009 | 100104 | 100148 | 100215 | 100220 | 100306 | 100413 | ... | 995678 | 995783 | 995915 | 995932 | 997094 | 997874 | 999920 | 999927 | 999939 | 999940 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **9** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

10 rows × 4589 columns

In [ ]:
```
print('Average number of products/order = ' + str(round(comparison_df.shape[0]/len(comparison_df.po_number.unique()), 1)))
```

```
Average number of products/order = 3.8
```

Since most orders only have a small subset of the full database of products, most of the values in the data frame are zeros. On average, each order only has 3.8 products.

**Support**

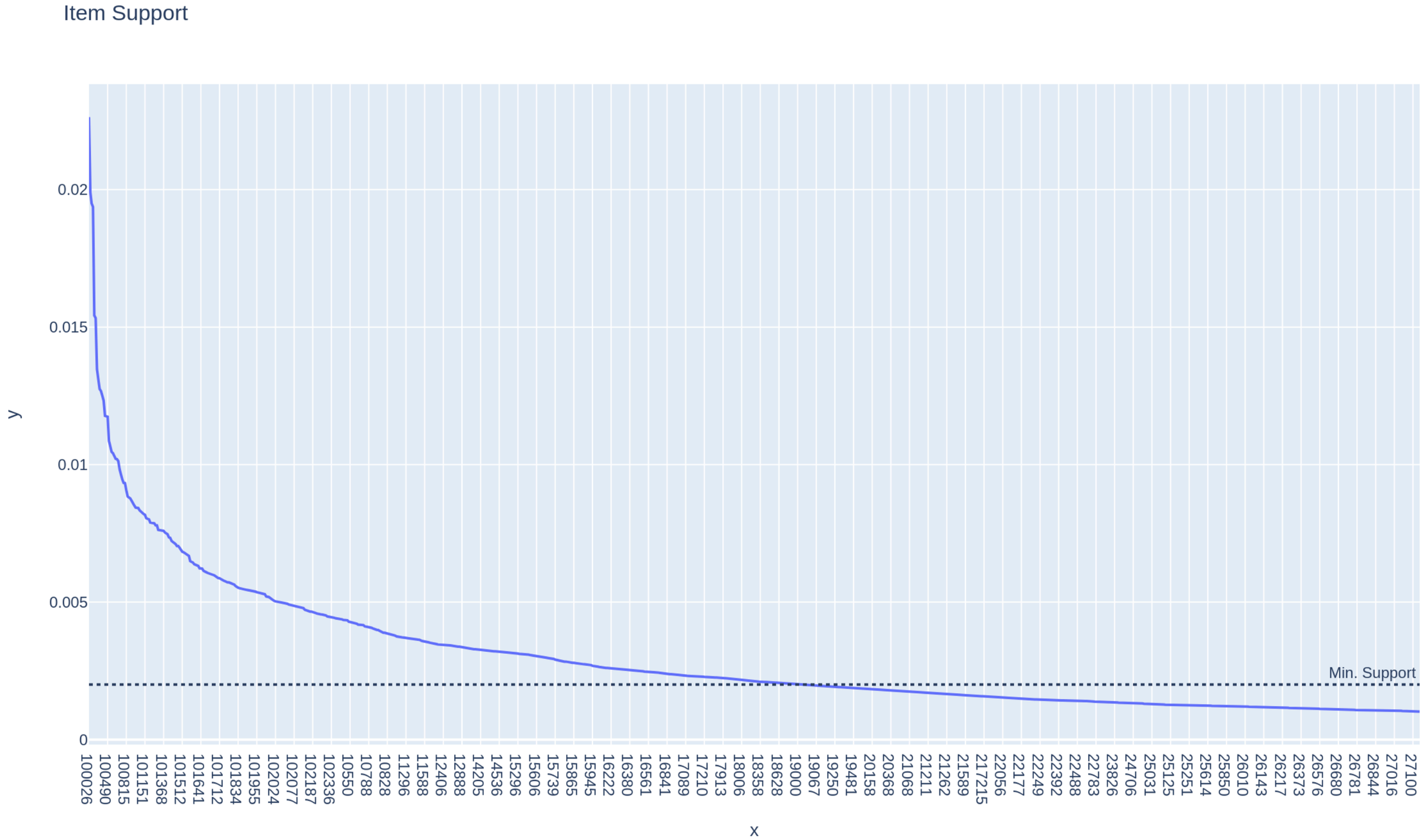
Support is a probability measure of a product's frequency in the database. In general, products with low support can be excluded from market basket analysis because they rarely appear in combinations that would give any meaningful insights. Setting a minimum support will eliminate unnecessary computation and memory consumption by excluding those products that are rarely seen.

In [ ]:
```
num_baskets = sparse_basket_df.shape[0]
item_frequency = [sparse_basket_df.loc[:,i].sum(axis=0)/num_baskets for i in sparse_basket_df.columns]
item_frequency.sort(reverse=True)
```

On exploring the chart below, a minimum support is set at 0.002

In [ ]:
```
fig= px.line([item_frequency], x= sparse_basket_df.columns[0:1000],
                               y= item_frequency[0:1000],
                    title='Item Support')

min_support = 0.002
fig.add_hline(y= min_support, line_dash="dot",
              annotation_text=f"Min. Support",
              annotation_position="top right")

Image(fig.to_image(format="png", width= 1200, height= 700,scale=2))
```

Out[ ]:




**Apriori Algorithm**

This is computationally efficient algorithm that uses a hashing scheme to find frequent item sets. While there are more effient methods like the FP-Growth algorithm, the implementation in this report does not require extensive memory.

In [ ]:
```
%time item_sets = apriori(sparse_basket_df, min_support=min_support, use_colnames=True, verbose=1, low_memory=True)
```

```
Processing 5 combinations | Sampling itemset size 4
CPU times: user 12.6 s, sys: 16.6 s, total: 29.2 s
Wall time: 29.4 s
```

In [ ]:
```
item_sets
```

Out[ ]:

| | support | itemsets |
|---|---|---|
| **0** | 0.004446 | (10008) |
| **1** | 0.011771 | (100413) |
| **2** | 0.005447 | (100423) |
| **3** | 0.010667 | (10548) |
| **4** | 0.007624 | (10550) |
| **...** | ... | ... |
| **841** | 0.002184 | (64863, 65013, 64870) |
| **842** | 0.003095 | (65013, 64864, 64870) |
| **843** | 0.002816 | (65013, 64865, 64870) |
| **844** | 0.002397 | (64904, 65013, 64870) |
| **845** | 0.002679 | (64858, 65013, 100413, 64870) |

846 rows × 2 columns

The next function creates named item sets by assigning the product_name for better readability

In [ ]:
```
def display_item_sets(set, product_names):
    prod_list = product_names.product_name
    named_sets =[]
    for item in set.itemsets:
        named_sets.append([prod_list[item_id] for item_id in list(item)])
    return pd.DataFrame({'support':set.support,
                         'itemsets':named_sets}).sort_values('support', ascending=False).reset_index()

pd.set_option('display.max_rows', None)
named_item_sets = display_item_sets(item_sets, product_names)
named_item_sets
```

Out[ ]:

| | index | support | itemsets |
|---|---|---|---|
| **0** | 32 | 0.022635 | [BLACK VELVET 1750ml (6 pack)] |
| **1** | 359 | 0.019901 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **2** | 356 | 0.019495 | [FIREBALL CINNAMON WHISKEY 100ml (48 pack)] |
| **3** | 197 | 0.019371 | [HAWKEYE VODKA 1750ml (6 pack)] |
| **4** | 233 | 0.015423 | [TITOS HANDMADE VODKA 750ml (12 pack)] |
| **5** | 349 | 0.015327 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **6** | 235 | 0.013456 | [TITOS HANDMADE VODKA 1750ml (6 pack)] |
| **7** | 234 | 0.013143 | [TITOS HANDMADE VODKA 1000ml (12 pack)] |
| **8** | 106 | 0.012748 | [SEAGRAMS 7 CROWN 1750ml (6 pack)] |
| **9** | 11 | 0.012676 | [CROWN ROYAL PEACH 750ml (12 pack)] |
| **10** | 263 | 0.012483 | [ADMIRAL NELSON SPICED 1750ml (6 pack)] |
| **11** | 329 | 0.012328 | [JOSE CUERVO AUTHENTIC LIME MARGARITA 1750ml (... |
| **12** | 1 | 0.011771 | [FIREBALL CINNAMON WHISKEY PARTY BUCKET 50ml (... |
| **13** | 200 | 0.011744 | [BLUE OX VODKA 1750ml (6 pack)] |
| **14** | 15 | 0.011740 | [CROWN ROYAL REGAL APPLE 750ml (12 pack)] |
| **15** | 792 | 0.010929 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **16** | 194 | 0.010853 | [HAWKEYE VODKA PET 750ml (12 pack)] |
| **17** | 3 | 0.010667 | [BLACK VELVET TOASTED CARAMEL 1750ml (6 pack)] |
| **18** | 20 | 0.010458 | [CROWN ROYAL 750ml (12 pack)] |
| **19** | 205 | 0.010413 | [MCCORMICK 80PRF VODKA PET 375ml (24 pack)] |
| **20** | 295 | 0.010272 | [PARAMOUNT WHITE RUM 1750ml (6 pack)] |
| **21** | 232 | 0.010210 | [TITOS HANDMADE VODKA 375ml (12 pack)] |
| **22** | 283 | 0.010196 | [CAPTAIN MORGAN ORIGINAL SPICED 1000ml (12 pack)] |
| **23** | 206 | 0.010128 | [MCCORMICK 80PRF VODKA PET 1750ml (6 pack)] |
| **24** | 256 | 0.009825 | [MALIBU COCONUT 750ml (12 pack)] |
| **25** | 114 | 0.009605 | [JACK DANIELS OLD #7 BLACK LABEL 750ml (12 pack)] |
| **26** | 31 | 0.009460 | [BLACK VELVET PET 750ml (12 pack)] |
| **27** | 770 | 0.009402 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **28** | 189 | 0.009337 | [FIVE O'CLOCK VODKA 1750ml (6 pack)] |
| **29** | 158 | 0.009326 | [ABSOLUT SWEDISH VODKA 80PRF 750ml (12 pack)] |
| **30** | 183 | 0.009072 | [BARTON VODKA 1750ml (6 pack)] |
| **31** | 282 | 0.008845 | [CAPTAIN MORGAN ORIGINAL SPICED 750ml (12 pack)] |
| **32** | 279 | 0.008800 | [CAPTAIN MORGAN ORIGINAL SPICED MINI 50ml (12 ... |
| **33** | 22 | 0.008769 | [CROWN ROYAL 1750ml (6 pack)] |
| **34** | 28 | 0.008683 | [BLACK VELVET 375ml (24 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **35** | 180 | 0.008549 | [SVEDKA 80PRF 1750ml (6 pack)] |
| **36** | 481 | 0.008511 | [OLE SMOKY SALTY CARAMEL WHISKEY 750ml (6 pack)] |
| **37** | 772 | 0.008494 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **38** | 352 | 0.008436 | [FIREBALL CINNAMON WHISKEY PET 750ml (12 pack)] |
| **39** | 229 | 0.008425 | [SMIRNOFF 80PRF PET 1750ml (6 pack)] |
| **40** | 169 | 0.008418 | [FRIS DANISH VODKA 1750ml (6 pack)] |
| **41** | 62 | 0.008336 | [CAPTAIN MORGAN ORIGINAL SPICED BARREL 1750ml ... |
| **42** | 231 | 0.008281 | [PLATINUM 7X VODKA 1750ml (6 pack)] |
| **43** | 116 | 0.008236 | [JACK DANIELS OLD #7 BLACK LABEL 1750ml (6 pack)] |
| **44** | 101 | 0.008202 | [FIVE STAR 1750ml (6 pack)] |
| **45** | 333 | 0.008174 | [JOSE CUERVO AUTHENTIC LIGHT LIME MARGARITA 17... |
| **46** | 324 | 0.008050 | [CHI-CHI'S MARGARITA 1750ml (6 pack)] |
| **47** | 82 | 0.008026 | [MAKERS MARK 750ml (12 pack)] |
| **48** | 61 | 0.008009 | [EVAN WILLIAMS BLACK 1750ml (6 pack)] |
| **49** | 351 | 0.007892 | [FIREBALL CINNAMON WHISKEY 375ml (24 pack)] |
| **50** | 237 | 0.007872 | [TITOS HANDMADE VODKA MINI 50ml (5 pack)] |
| **51** | 38 | 0.007868 | [WINDSOR CANADIAN PET 1750ml (6 pack)] |
| **52** | 112 | 0.007865 | [JACK DANIELS OLD #7 BLACK LABEL MINI 50ml (12... |
| **53** | 357 | 0.007793 | [FIREBALL CINNAMON WHISKEY PET 1750ml (6 pack)] |
| **54** | 74 | 0.007789 | [JIM BEAM 750ml (12 pack)] |
| **55** | 4 | 0.007624 | [BLACK VELVET TOASTED CARAMEL 750ml (12 pack)] |
| **56** | 219 | 0.007617 | [ROW VODKA 1750ml (6 pack)] |
| **57** | 510 | 0.007610 | [PATRON SILVER 750ml (12 pack)] |
| **58** | 41 | 0.007590 | [JAMESON 750ml (12 pack)] |
| **59** | 30 | 0.007590 | [BLACK VELVET 1000ml (12 pack)] |
| **60** | 92 | 0.007531 | [TEN HIGH 1750ml (6 pack)] |
| **61** | 215 | 0.007514 | [PHILLIPS VODKA 1750ml (6 pack)] |
| **62** | 495 | 0.007466 | [JOSE CUERVO ESPECIAL SILVER 750ml (12 pack)] |
| **63** | 56 | 0.007352 | [BULLEIT BOURBON 750ml (12 pack)] |
| **64** | 193 | 0.007335 | [HAWKEYE VODKA 375ml (24 pack)] |
| **65** | 21 | 0.007222 | [CROWN ROYAL 1000ml (12 pack)] |
| **66** | 115 | 0.007187 | [JACK DANIELS OLD #7 BLACK LABEL 1000ml (12 pa... |
| **67** | 71 | 0.007149 | [JIM BEAM MINI 50ml (12 pack)] |
| **68** | 293 | 0.007112 | [PARAMOUNT GOLD RUM 1750ml (6 pack)] |
| **69** | 647 | 0.007081 | [ADMIRAL NELSON SPICED 1750ml (6 pack), BLACK ... |

| | index | support | itemsets |
|---|---|---|---|
| **70** | 29 | 0.007039 | [BLACK VELVET 750ml (12 pack)] |
| **71** | 415 | 0.007039 | [THE ORIGINAL PICKLE SHOT DILL PICKLE VODKA 75... |
| **72** | 67 | 0.006978 | [FOUR ROSES BOURBON 750ml (12 pack)] |
| **73** | 258 | 0.006895 | [MALIBU COCONUT 1750ml (6 pack)] |
| **74** | 548 | 0.006881 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **75** | 557 | 0.006840 | [BLACK VELVET TOASTED CARAMEL 1750ml (6 pack),... |
| **76** | 276 | 0.006833 | [CAPTAIN MORGAN ORIGINAL SPICED PET 750ml (12 ... |
| **77** | 6 | 0.006812 | [CANADIAN CLUB WHISKY 1750ml (6 pack)] |
| **78** | 225 | 0.006788 | [SMIRNOFF 80PRF 375ml (24 pack)] |
| **79** | 12 | 0.006733 | [CROWN ROYAL REGAL APPLE MINI 50ml (10 pack)] |
| **80** | 223 | 0.006723 | [SMIRNOFF 80PRF MINI 50ml (12 pack)] |
| **81** | 523 | 0.006678 | [JOSE CUERVO ESPECIAL REPOSADO 750ml (12 pack)] |
| **82** | 546 | 0.006555 | [FIREBALL CINNAMON WHISKEY PARTY BUCKET 50ml (... |
| **83** | 511 | 0.006486 | [TEREMANA BLANCO TEQUILA 750ml (6 pack)] |
| **84** | 447 | 0.006465 | [PARAMOUNT PEPPERMINT SCHNAPPS 1750ml (6 pack)] |
| **85** | 236 | 0.006438 | [TITOS HANDMADE VODKA 200ml (24 pack)] |
| **86** | 736 | 0.006389 | [TITOS HANDMADE VODKA 1750ml (6 pack), TITOS H... |
| **87** | 203 | 0.006376 | [BARTON NATURALS VODKA 1750ml (6 pack)] |
| **88** | 177 | 0.006362 | [PINNACLE 1750ml (6 pack)] |
| **89** | 218 | 0.006345 | [ROW VODKA 750ml (12 pack)] |
| **90** | 64 | 0.006314 | [FOUR ROSES SMALL BATCH 750ml (6 pack)] |
| **91** | 301 | 0.006228 | [HENNESSY VS 750ml (12 pack)] |
| **92** | 281 | 0.006228 | [CAPTAIN MORGAN ORIGINAL SPICED 375ml (24 pack)] |
| **93** | 270 | 0.006214 | [BACARDI SUPERIOR 1000ml (12 pack)] |
| **94** | 368 | 0.006135 | [LICOR 43 750ml (12 pack)] |
| **95** | 257 | 0.006128 | [MALIBU COCONUT 1000ml (12 pack)] |
| **96** | 353 | 0.006087 | [FIREBALL CINNAMON WHISKEY 750ml (12 pack)] |
| **97** | 24 | 0.006059 | [SEAGRAMS VO CANADIAN WHISKEY PET 1750ml (6 pa... |
| **98** | 156 | 0.006049 | [ABSOLUT SWEDISH VODKA 80PRF MINI 50ml (10 pack)] |
| **99** | 222 | 0.006025 | [SKOL VODKA 1750ml (6 pack)] |
| **100** | 19 | 0.006004 | [CROWN ROYAL 375ml (24 pack)] |
| **101** | 174 | 0.005984 | [KETEL ONE 750ml (12 pack)] |
| **102** | 466 | 0.005973 | [DEKUYPER TRIPLE SEC 1000ml (12 pack)] |
| **103** | 60 | 0.005942 | [EVAN WILLIAMS BLACK 750ml (12 pack)] |
| **104** | 173 | 0.005881 | [GREY GOOSE 750ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **105** | 250 | 0.005874 | [UV BLUE RASPBERRY 1750ml (6 pack)] |
| **106** | 14 | 0.005867 | [CROWN ROYAL REGAL APPLE 375ml (24 pack)] |
| **107** | 839 | 0.005850 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **108** | 265 | 0.005829 | [BACARDI SUPERIOR PET 1750ml (6 pack)] |
| **109** | 380 | 0.005819 | [BAILEYS ORIGINAL IRISH CREAM 750ml (12 pack)] |
| **110** | 377 | 0.005781 | [KAHLUA COFFEE 750ml (12 pack)] |
| **111** | 228 | 0.005764 | [SMIRNOFF 80PRF PET 750ml (12 pack)] |
| **112** | 702 | 0.005746 | [HAWKEYE VODKA PET 750ml (12 pack), HAWKEYE VO... |
| **113** | 467 | 0.005729 | [JUAREZ TRIPLE SEC 1000ml (12 pack)] |
| **114** | 102 | 0.005715 | [KESSLER BLEND WHISKEY 1750ml (6 pack)] |
| **115** | 73 | 0.005715 | [JIM BEAM 375ml (24 pack)] |
| **116** | 388 | 0.005698 | [DR MCGILLICUDDYS CHERRY 1750ml (6 pack)] |
| **117** | 354 | 0.005695 | [FIREBALL CINNAMON WHISKEY 1000ml (12 pack)] |
| **118** | 397 | 0.005681 | [RUMCHATA 750ml (12 pack)] |
| **119** | 59 | 0.005636 | [ELIJAH CRAIG 750ml (12 pack)] |
| **120** | 572 | 0.005592 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **121** | 386 | 0.005585 | [DR MCGILLICUDDYS CHERRY 750ml (12 pack)] |
| **122** | 590 | 0.005561 | [CROWN ROYAL REGAL APPLE 750ml (12 pack), CROW... |
| **123** | 332 | 0.005533 | [JOSE CUERVO AUTHENTIC STRAWBERRY LIME MARGARI... |
| **124** | 350 | 0.005523 | [FIREBALL CINNAMON WHISKEY 200ml (48 pack)] |
| **125** | 195 | 0.005506 | [HAWKEYE VODKA 750ml (12 pack)] |
| **126** | 226 | 0.005488 | [SMIRNOFF 80PRF 750ml (12 pack)] |
| **127** | 300 | 0.005482 | [HENNESSY VS 375ml (12 pack)] |
| **128** | 18 | 0.005451 | [CROWN ROYAL MINI 50ml (10 pack)] |
| **129** | 2 | 0.005447 | [FIREBALL CINNAMON WHISKEY 100ML CARRIER 100ml... |
| **130** | 454 | 0.005447 | [DEKUYPER LUSCIOUS PEACHTREE 1000ml (12 pack)] |
| **131** | 211 | 0.005437 | [NIKOLAI VODKA 1750ml (6 pack)] |
| **132** | 190 | 0.005430 | [FIVE O'CLOCK VODKA PET 750ml (12 pack)] |
| **133** | 343 | 0.005420 | [DISARONNO AMARETTO 750ml (12 pack)] |
| **134** | 138 | 0.005416 | [TANQUERAY GIN 750ml (12 pack)] |
| **135** | 16 | 0.005413 | [CROWN ROYAL REGAL APPLE 1000ml (12 pack)] |
| **136** | 735 | 0.005406 | [TITOS HANDMADE VODKA 1000ml (12 pack), TITOS ... |
| **137** | 86 | 0.005396 | [OLD CROW 1750ml (6 pack)] |
| **138** | 58 | 0.005378 | [CEDAR RIDGE BOURBON 750ml (6 pack)] |
| **139** | 26 | 0.005358 | [BLACK VELVET MINI 50ml (8 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **140** | 244 | 0.005347 | [DEEP EDDY LEMON 750ml (12 pack)] |
| **141** | 271 | 0.005323 | [BACARDI SUPERIOR 1750ml (6 pack)] |
| **142** | 95 | 0.005306 | [WOODFORD RESERVE 750ml (12 pack)] |
| **143** | 393 | 0.005292 | [RUMPLE MINZE PEPPERMINT SCHNAPPS LIQUEUR 1000... |
| **144** | 27 | 0.005289 | [BLACK VELVET 200ml (48 pack)] |
| **145** | 465 | 0.005286 | [BLACK VELVET PEACH CANADIAN WHISKEY 1750ml (6... |
| **146** | 710 | 0.005265 | [PARAMOUNT WHITE RUM 1750ml (6 pack), HAWKEYE ... |
| **147** | 160 | 0.005203 | [ABSOLUT SWEDISH VODKA 80PRF 1750ml (6 pack)] |
| **148** | 75 | 0.005203 | [JIM BEAM 1000ml (12 pack)] |
| **149** | 266 | 0.005182 | [BACARDI SUPERIOR MINI 50ml (12 pack)] |
| **150** | 348 | 0.005148 | [COINTREAU LIQUEUR 750ml (12 pack)] |
| **151** | 119 | 0.005096 | [TEMPLETON RYE 4YR 750ml (6 pack)] |
| **152** | 543 | 0.005093 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **153** | 498 | 0.005090 | [DON JULIO BLANCO 750ml (6 pack)] |
| **154** | 33 | 0.005041 | [CANADIAN LTD WHISKY 1750ml (6 pack)] |
| **155** | 355 | 0.005028 | [FIREBALL CINNAMON WHISKEY 1750ml (6 pack)] |
| **156** | 472 | 0.005021 | [BLACK VELVET APPLE 1750ml (6 pack)] |
| **157** | 42 | 0.005017 | [JAMESON 1000ml (12 pack)] |
| **158** | 639 | 0.005007 | [BLACK VELVET PET 750ml (12 pack), BLACK VELVE... |
| **159** | 125 | 0.005007 | [RED STAG BLACK CHERRY 750ml (12 pack)] |
| **160** | 492 | 0.005004 | [CASAMIGOS BLANCO 750ml (6 pack)] |
| **161** | 17 | 0.004986 | [CROWN ROYAL REGAL APPLE 1750ml (6 pack)] |
| **162** | 269 | 0.004976 | [BACARDI SUPERIOR 750ml (12 pack)] |
| **163** | 198 | 0.004966 | [BLUE OX VODKA 750ml (12 pack)] |
| **164** | 209 | 0.004955 | [NEW AMSTERDAM 80PRF 750ml (12 pack)] |
| **165** | 574 | 0.004949 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **166** | 240 | 0.004938 | [NEW AMSTERDAM PINK WHITNEY MINI 50ml (12 pack)] |
| **167** | 365 | 0.004911 | [JAGERMEISTER LIQUEUR 750ml (12 pack)] |
| **168** | 241 | 0.004900 | [NEW AMSTERDAM PINK WHITNEY 750ml (12 pack)] |
| **169** | 363 | 0.004883 | [TEQUILA ROSE LIQUEUR MINI 50ml (6 pack)] |
| **170** | 499 | 0.004856 | [1800 SILVER 750ml (12 pack)] |
| **171** | 79 | 0.004852 | [KNOB CREEK 750ml (6 pack)] |
| **172** | 153 | 0.004845 | [REVELTON MULBERRY GIN 750ml (6 pack)] |
| **173** | 369 | 0.004842 | [JAMESON ORANGE 750ml (6 pack)] |
| **174** | 391 | 0.004835 | [RUMPLE MINZE PEPPERMINT SCHNAPPS LIQUEUR MINI... |

| | index | support | itemsets |
|---|---|---|---|
| **175** | 207 | 0.004804 | [[NEW AMSTERDAM 80PRF MINI 50ml (10 pack), NEW... |
| **176** | 170 | 0.004804 | [GREY GOOSE 1000ml (6 pack)] |
| **177** | 520 | 0.004787 | [JOSE CUERVO ESPECIAL REPOSADO MINI 50ml (12 p... |
| **178** | 513 | 0.004780 | [HORNITOS PLATA 750ml (12 pack)] |
| **179** | 779 | 0.004753 | [FIREBALL CINNAMON WHISKEY 375ml (24 pack), FI... |
| **180** | 25 | 0.004722 | [BLACK VELVET RESERVE 1750ml (6 pack)] |
| **181** | 196 | 0.004701 | [HAWKEYE VODKA 1000ml (12 pack)] |
| **182** | 8 | 0.004673 | [CROWN ROYAL PEACH 1750ml (6 pack)] |
| **183** | 322 | 0.004660 | [CHI-CHI'S GOLD MARGARITA 1750ml (6 pack)] |
| **184** | 323 | 0.004653 | [CHI-CHI'S MEXICAN MUDSLIDE 1750ml (6 pack)] |
| **185** | 213 | 0.004653 | [UV VODKA PET 1750ml (6 pack)] |
| **186** | 372 | 0.004625 | [ST GERMAIN 750ml (6 pack)] |
| **187** | 470 | 0.004622 | [SKREWBALL PEANUT BUTTER WHISKEY 750ml (12 pack)] |
| **188** | 191 | 0.004605 | [FLEISCHMANNS 80PRF VODKA 1750ml (6 pack)] |
| **189** | 192 | 0.004584 | [HAWKEYE VODKA MINI 50ml (12 pack)] |
| **190** | 464 | 0.004584 | [DEKUYPER BLUE CURACAO 750ml (12 pack)] |
| **191** | 442 | 0.004581 | [RYANS CREAM LIQUEUR 1750ml (6 pack)] |
| **192** | 134 | 0.004553 | [BOMBAY SAPPHIRE GIN 750ml (12 pack)] |
| **193** | 780 | 0.004546 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **194** | 433 | 0.004546 | [BUFFALO TRACE BOURBON CREAM 750ml (12 pack)] |
| **195** | 10 | 0.004539 | [CROWN ROYAL VANILLA 750ml (12 pack)] |
| **196** | 139 | 0.004515 | [TANQUERAY GIN 1000ml (12 pack)] |
| **197** | 334 | 0.004512 | [JOSE CUERVO GOLDEN MARGARITA 1750ml (6 pack)] |
| **198** | 617 | 0.004491 | [CROWN ROYAL 1000ml (12 pack), CAPTAIN MORGAN ... |
| **199** | 395 | 0.004477 | [RUMCHATA MINI 50ml (12 pack)] |
| **200** | 137 | 0.004474 | [HENDRICKS GIN 750ml (6 pack)] |
| **201** | 817 | 0.004457 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **202** | 440 | 0.004453 | [OLE SMOKY WHITE CHOCOLATE STRAWBERRY MOONSHIN... |
| **203** | 644 | 0.004453 | [EVAN WILLIAMS BLACK 1750ml (6 pack), BLACK VE... |
| **204** | 0 | 0.004446 | [SCORESBY RARE SCOTCH 1750ml (6 pack)] |
| **205** | 784 | 0.004446 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **206** | 105 | 0.004443 | [SEAGRAMS 7 CROWN 1000ml (12 pack)] |
| **207** | 487 | 0.004429 | [OLE SMOKY SALTY WATERMELON WHISKEY 750ml (6 p... |
| **208** | 761 | 0.004422 | [JOSE CUERVO AUTHENTIC LIGHT LIME MARGARITA 17... |
| **209** | 474 | 0.004412 | [JACK DANIELS TENNESSEE HONEY 750ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **210** | 366 | 0.004409 | [JAGERMEISTER LIQUEUR 1000ml (12 pack)] |
| **211** | 253 | 0.004398 | [EVERCLEAR 151 750ml (12 pack)] |
| **212** | 201 | 0.004398 | [MCCORMICK 80PRF VODKA 750ml (12 pack)] |
| **213** | 399 | 0.004378 | [DI AMORE AMARETTO 750ml (12 pack)] |
| **214** | 373 | 0.004371 | [GRANGALA TRIPLE ORANGE LIQUEUR 750ml (12 pack)] |
| **215** | 696 | 0.004354 | [BARTON VODKA 1750ml (6 pack), FIREBALL CINNAM... |
| **216** | 94 | 0.004350 | [WILD TURKEY 101 750ml (12 pack)] |
| **217** | 77 | 0.004347 | [JIM BEAM PET 1750ml (6 pack)] |
| **218** | 387 | 0.004336 | [DR MCGILLICUDDYS CHERRY 1000ml (12 pack)] |
| **219** | 136 | 0.004336 | [JIM BEAM PEACH 750ml (12 pack)] |
| **220** | 739 | 0.004336 | [TITOS HANDMADE VODKA 1750ml (6 pack), TITOS H... |
| **221** | 490 | 0.004288 | [OLE SMOKY PEANUT BUTTER WHISKEY 750ml (6 pack)] |
| **222** | 326 | 0.004278 | [CHI-CHI'S SKINNY MARGARITA 1750ml (6 pack)] |
| **223** | 782 | 0.004278 | [FIREBALL CINNAMON WHISKEY PET 750ml (12 pack)... |
| **224** | 579 | 0.004264 | [CROWN ROYAL PEACH 750ml (12 pack), SEAGRAMS 7... |
| **225** | 426 | 0.004261 | [SMIRNOFF VANILLA 750ml (12 pack)] |
| **226** | 344 | 0.004257 | [APEROL 750ml (6 pack)] |
| **227** | 150 | 0.004257 | [PARAMOUNT GIN 1750ml (6 pack)] |
| **228** | 76 | 0.004254 | [JIM BEAM 1750ml (6 pack)] |
| **229** | 39 | 0.004213 | [FINAGRENS IRISH WHISKEY 750ml (12 pack)] |
| **230** | 633 | 0.004195 | [BLACK VELVET 375ml (24 pack), BLACK VELVET 17... |
| **231** | 261 | 0.004182 | [ADMIRAL NELSON SPICED PET 750ml (12 pack)] |
| **232** | 508 | 0.004175 | [LAUDERS 1750ml (6 pack)] |
| **233** | 182 | 0.004165 | [BARTON VODKA 1000ml (12 pack)] |
| **234** | 53 | 0.004165 | [BASIL HAYDEN 8YR 750ml (12 pack)] |
| **235** | 63 | 0.004161 | [BUFFALO TRACE BOURBON 750ml (12 pack)] |
| **236** | 688 | 0.004123 | [FRIS DANISH VODKA 1750ml (6 pack), FIREBALL C... |
| **237** | 438 | 0.004116 | [OLE SMOKY WHITE CHOCOLATE STRAWBERRY CREAM 75... |
| **238** | 87 | 0.004113 | [OLD FORESTER BOURBON 86PRF 750ml (12 pack)] |
| **239** | 147 | 0.004092 | [GORDONS GIN LONDON DRY PET 1750ml (6 pack)] |
| **240** | 316 | 0.004089 | [JOHNNIE WALKER RED 750ml (12 pack)] |
| **241** | 730 | 0.004082 | [TITOS HANDMADE VODKA 375ml (12 pack), TITOS H... |
| **242** | 525 | 0.004075 | [JOSE CUERVO ESPECIAL REPOSADO 1750ml (6 pack)] |
| **243** | 385 | 0.004065 | [DR MCGILLICUDDYS CHERRY MINI 50ml (12 pack)] |
| **244** | 503 | 0.004044 | [JUAREZ SILVER 1000ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **245** | 453 | 0.004013 | [DEKUYPER LUSCIOUS PEACHTREE 750ml (12 pack)] |
| **246** | 486 | 0.004006 | [SOUTHERN COMFORT PET 1750ml (6 pack)] |
| **247** | 563 | 0.003999 | [BLACK VELVET 1750ml (6 pack), BLACK VELVET TO... |
| **248** | 113 | 0.003989 | [JACK DANIELS OLD #7 BLACK LABEL 200ml (48 pack)] |
| **249** | 729 | 0.003986 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **250** | 595 | 0.003982 | [CROWN ROYAL REGAL APPLE 750ml (12 pack), SEAG... |
| **251** | 524 | 0.003982 | [JOSE CUERVO ESPECIAL REPOSADO 1000ml (12 pack)] |
| **252** | 559 | 0.003979 | [BLACK VELVET TOASTED CARAMEL 1750ml (6 pack),... |
| **253** | 635 | 0.003975 | [BLACK VELVET 750ml (12 pack), BLACK VELVET 17... |
| **254** | 437 | 0.003955 | [OLE SMOKY BANANA PUDDING CREAM MOONSHINE 750m... |
| **255** | 802 | 0.003951 | [FIREBALL CINNAMON WHISKEY PET 1750ml (6 pack)... |
| **256** | 176 | 0.003944 | [KETEL ONE 1750ml (6 pack)] |
| **257** | 695 | 0.003938 | [BARTON VODKA 1750ml (6 pack), FIREBALL CINNAM... |
| **258** | 306 | 0.003896 | [CHRISTIAN BROS BRANDY 750ml (12 pack)] |
| **259** | 163 | 0.003886 | [ABSOLUT CITRON 750ml (12 pack)] |
| **260** | 297 | 0.003883 | [KRAKEN BLACK SPICED RUM 750ml (12 pack)] |
| **261** | 122 | 0.003872 | [JIM BEAM HONEY 750ml (12 pack)] |
| **262** | 757 | 0.003869 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **263** | 791 | 0.003862 | [FIREBALL CINNAMON WHISKEY PET 1750ml (6 pack)... |
| **264** | 129 | 0.003845 | [JIM BEAM APPLE MINI 50ml (12 pack)] |
| **265** | 338 | 0.003821 | [1800 ULTIMATE MARGARITA PET 1750ml (6 pack)] |
| **266** | 311 | 0.003814 | [E & J VS 750ml (12 pack)] |
| **267** | 145 | 0.003810 | [GILBEYS GIN 1750ml (6 pack)] |
| **268** | 403 | 0.003810 | [[KINKY PINK MINI 50ml (6 pack), KINKY PINK MI... |
| **269** | 584 | 0.003807 | [CROWN ROYAL PEACH 750ml (12 pack), CAPTAIN MO... |
| **270** | 370 | 0.003803 | [LICOR 43 CHOCOLATE 750ml (6 pack)] |
| **271** | 687 | 0.003800 | [FRIS DANISH VODKA 1750ml (6 pack), FIREBALL C... |
| **272** | 13 | 0.003779 | [CROWN ROYAL REGAL APPLE 200ml (44 pack)] |
| **273** | 709 | 0.003776 | [PARAMOUNT GOLD RUM 1750ml (6 pack), HAWKEYE V... |
| **274** | 409 | 0.003748 | [OLE SMOKY BUTTER PECAN MOONSHINE 750ml (6 pack)] |
| **275** | 307 | 0.003748 | [CHRISTIAN BROS BRANDY 1750ml (6 pack)] |
| **276** | 552 | 0.003745 | [FIREBALL CINNAMON WHISKEY 100ML CARRIER 100ml... |
| **277** | 480 | 0.003738 | [OLE SMOKY MANGO HABANERO WHISKEY 750ml (6 pack)] |
| **278** | 482 | 0.003731 | [SOUTHERN COMFORT MINI 50ml (10 pack)] |
| **279** | 159 | 0.003714 | [ABSOLUT SWEDISH VODKA 80PRF 1000ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **280** | 422 | 0.003711 | [SMIRNOFF RASPBERRY 750ml (12 pack)] |
| **281** | 756 | 0.003704 | [CHI-CHI'S MARGARITA 1750ml (6 pack), FIREBALL... |
| **282** | 142 | 0.003700 | [AVIATION AMERICAN GIN 750ml (6 pack)] |
| **283** | 208 | 0.003700 | [NEW AMSTERDAM 80PRF 375ml (24 pack)] |
| **284** | 149 | 0.003697 | [NEW AMSTERDAM GIN 750ml (12 pack)] |
| **285** | 88 | 0.003686 | [REBEL KSBW 100PRF 750ml (6 pack)] |
| **286** | 299 | 0.003683 | [HENNESSY VS 200ml (24 pack)] |
| **287** | 717 | 0.003680 | [MCCORMICK 80PRF VODKA PET 1750ml (6 pack), MC... |
| **288** | 417 | 0.003676 | [THE ORIGINAL PICKLE SHOT SPICY PICKLE VODKA 7... |
| **289** | 509 | 0.003673 | [PATRON SILVER 375ml (12 pack)] |
| **290** | 576 | 0.003673 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **291** | 289 | 0.003659 | [CRUZAN BLACK CHERRY 750ml (12 pack)] |
| **292** | 435 | 0.003642 | [IOWISH CREAM SALTED CARAMEL 750ml (6 pack)] |
| **293** | 420 | 0.003638 | [SMIRNOFF KISSED CARAMEL 750ml (12 pack)] |
| **294** | 52 | 0.003631 | [ANGELS ENVY BOURBON PORT FINISHED 750ml (6 pa... |
| **295** | 140 | 0.003628 | [TANQUERAY GIN 1750ml (6 pack)] |
| **296** | 130 | 0.003625 | [JIM BEAM APPLE 750ml (12 pack)] |
| **297** | 815 | 0.003621 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **298** | 328 | 0.003587 | [JOSE CUERVO AUTHENTIC LIME MARGARITA 800ml (6... |
| **299** | 111 | 0.003576 | [JACK DANIELS OLD #7 BLACK LABEL FLAT 375ml (2... |
| **300** | 512 | 0.003559 | [TEREMANA REPOSADO TEQUILA 750ml (6 pack)] |
| **301** | 676 | 0.003556 | [CAPTAIN MORGAN ORIGINAL SPICED 1000ml (12 pac... |
| **302** | 57 | 0.003535 | [CABIN STILL STR BOURBON 1000ml (12 pack)] |
| **303** | 124 | 0.003535 | [BIRD DOG BLACKBERRY 750ml (6 pack)] |
| **304** | 412 | 0.003535 | [PARAMOUNT AMARETTO 1000ml (12 pack)] |
| **305** | 609 | 0.003535 | [SEAGRAMS 7 CROWN 1750ml (6 pack), CROWN ROYAL... |
| **306** | 664 | 0.003528 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **307** | 551 | 0.003521 | [FIREBALL CINNAMON WHISKEY 100ML CARRIER 100ml... |
| **308** | 423 | 0.003521 | [WILD TURKEY AMERICAN HONEY 750ml (12 pack)] |
| **309** | 473 | 0.003518 | [JACK DANIELS TENNESSEE HONEY MINI 50ml (12 pa... |
| **310** | 816 | 0.003511 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **311** | 777 | 0.003504 | [FIREBALL CINNAMON WHISKEY 200ml (48 pack), FI... |
| **312** | 507 | 0.003497 | [OLMECA ALTOS PLATA 750ml (6 pack)] |
| **313** | 694 | 0.003494 | [BARTON VODKA 1750ml (6 pack), FIREBALL CINNAM... |
| **314** | 135 | 0.003487 | [JIM BEAM PEACH MINI 50ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **315** | 260 | 0.003484 | [ADMIRAL NELSON SPICED 375ml (24 pack)] |
| **316** | 144 | 0.003470 | [FLEISCHMANNS GIN 1750ml (6 pack)] |
| **317** | 230 | 0.003456 | [STATE VODKA 1750ml (6 pack)] |
| **318** | 37 | 0.003456 | [PENDLETON CANADIAN WHISKEY 750ml (12 pack)] |
| **319** | 767 | 0.003446 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **320** | 416 | 0.003446 | [OLE SMOKY COOKIE DOUGH WHISKEY 750ml (6 pack)] |
| **321** | 165 | 0.003432 | [ABSOLUT RASPBERRI 750ml (12 pack)] |
| **322** | 345 | 0.003432 | [BUMBU RUM 750ml (6 pack)] |
| **323** | 515 | 0.003432 | [CLASE AZUL REPOSADO 750ml (6 pack)] |
| **324** | 601 | 0.003429 | [CROWN ROYAL 1000ml (12 pack), CROWN ROYAL REG... |
| **325** | 477 | 0.003429 | [OLE SMOKY APPLE PIE MOONSHINE 70PRF MINI 50ml... |
| **326** | 84 | 0.003429 | [MAKERS MARK 1750ml (6 pack)] |
| **327** | 404 | 0.003429 | [KINKY PINK 750ml (6 pack)] |
| **328** | 277 | 0.003425 | [CAPTAIN MORGAN PRIVATE STOCK 750ml (12 pack)] |
| **329** | 731 | 0.003422 | [TITOS HANDMADE VODKA 375ml (12 pack), TITOS H... |
| **330** | 44 | 0.003418 | [JAMESON 375ml (24 pack)] |
| **331** | 701 | 0.003411 | [HAWKEYE VODKA 375ml (24 pack), HAWKEYE VODKA ... |
| **332** | 728 | 0.003405 | [PLATINUM 7X VODKA 1750ml (6 pack), FIREBALL C... |
| **333** | 592 | 0.003405 | [CROWN ROYAL REGAL APPLE 750ml (12 pack), CROW... |
| **334** | 262 | 0.003405 | [ADMIRAL NELSON SPICED 750ml (12 pack)] |
| **335** | 117 | 0.003398 | [BULLEIT 95 RYE 750ml (12 pack)] |
| **336** | 522 | 0.003398 | [JOSE CUERVO ESPECIAL REPOSADO FLASK 375ml (24... |
| **337** | 242 | 0.003384 | [DEEP EDDY LEMON MINI 50ml (12 pack)] |
| **338** | 255 | 0.003380 | [MALIBU PINEAPPLE 750ml (12 pack)] |
| **339** | 619 | 0.003377 | [CROWN ROYAL 1750ml (6 pack), SEAGRAMS 7 CROWN... |
| **340** | 637 | 0.003377 | [BLACK VELVET 1750ml (6 pack), BLACK VELVET 10... |
| **341** | 521 | 0.003377 | [JOSE CUERVO ESPECIAL REPOSADO 200ml (48 pack)] |
| **342** | 602 | 0.003374 | [CAPTAIN MORGAN ORIGINAL SPICED 1000ml (12 pac... |
| **343** | 768 | 0.003370 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **344** | 600 | 0.003367 | [CROWN ROYAL REGAL APPLE 750ml (12 pack), CAPT... |
| **345** | 485 | 0.003367 | [SOUTHERN COMFORT 1000ml (12 pack)] |
| **346** | 294 | 0.003363 | [PARAMOUNT WHITE RUM 1000ml (12 pack)] |
| **347** | 248 | 0.003356 | [NEW AMSTERDAM PEACH 750ml (12 pack)] |
| **348** | 783 | 0.003346 | [FIREBALL CINNAMON WHISKEY PET 1750ml (6 pack)... |
| **349** | 346 | 0.003325 | [CARAVELLA LIMONCELLO 750ml (6 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **350** | 268 | 0.003322 | [BACARDI SUPERIOR PET 750ml (12 pack)] |
| **351** | 148 | 0.003315 | [NEW AMSTERDAM GIN 1750ml (6 pack)] |
| **352** | 501 | 0.003312 | [ESPOLON BLANCO 750ml (12 pack)] |
| **353** | 96 | 0.003301 | [WOODFORD RESERVE DOUBLE OAKED 750ml (6 pack)] |
| **354** | 392 | 0.003298 | [RUMPLE MINZE PEPPERMINT SCHNAPPS LIQUEUR 750m... |
| **355** | 408 | 0.003294 | [OLE SMOKY BUTTER PECAN MOONSHINE MINI 50ml (8... |
| **356** | 628 | 0.003291 | [BLACK VELVET 1750ml (6 pack), BLACK VELVET RE... |
| **357** | 204 | 0.003284 | [MCCORMICK 80PRF VODKA 200ml (48 pack)] |
| **358** | 678 | 0.003284 | [JACK DANIELS OLD #7 BLACK LABEL 1750ml (6 pac... |
| **359** | 273 | 0.003284 | [BACARDI LIMON 1000ml (12 pack)] |
| **360** | 484 | 0.003277 | [SOUTHERN COMFORT 750ml (12 pack)] |
| **361** | 103 | 0.003270 | [SEAGRAMS 7 CROWN 375ml (12 pack)] |
| **362** | 221 | 0.003267 | [SEAGRAMS EXTRA SMOOTH VODKA 1750ml (6 pack)] |
| **363** | 461 | 0.003257 | [[99 BANANAS MINI 50ml (1 pack), 99 BANANAS MI... |
| **364** | 733 | 0.003253 | [TITOS HANDMADE VODKA 200ml (24 pack), TITOS H... |
| **365** | 686 | 0.003253 | [FRIS DANISH VODKA 1750ml (6 pack), FIREBALL C... |
| **366** | 441 | 0.003243 | [RYANS CREAM LIQUEUR 750ml (12 pack)] |
| **367** | 83 | 0.003233 | [MAKERS MARK 1000ml (12 pack)] |
| **368** | 249 | 0.003229 | [UV BLUE RASPBERRY MINI 50ml (12 pack)] |
| **369** | 643 | 0.003229 | [BLACK VELVET 1750ml (6 pack), EVAN WILLIAMS B... |
| **370** | 530 | 0.003219 | [HORNITOS REPOSADO 750ml (12 pack)] |
| **371** | 439 | 0.003219 | [OLE SMOKY BANANA PUDDING CREAM MOONSHINE MINI... |
| **372** | 296 | 0.003215 | [SAILOR JERRY SPICED NAVY RUM 750ml (12 pack)] |
| **373** | 661 | 0.003215 | [CAPTAIN MORGAN ORIGINAL SPICED BARREL 1750ml ... |
| **374** | 425 | 0.003215 | [SMIRNOFF STRAWBERRY 750ml (12 pack)] |
| **375** | 315 | 0.003215 | [JOHNNIE WALKER BLACK 750ml (12 pack)] |
| **376** | 820 | 0.003215 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **377** | 626 | 0.003215 | [SEAGRAMS VO CANADIAN WHISKEY PET 1750ml (6 pa... |
| **378** | 274 | 0.003215 | [BACARDI LIMON 1750ml (6 pack)] |
| **379** | 649 | 0.003212 | [BLACK VELVET PEACH CANADIAN WHISKEY 1750ml (6... |
| **380** | 378 | 0.003212 | [KAHLUA COFFEE 1000ml (12 pack)] |
| **381** | 451 | 0.003212 | [DEKUYPER BUTTERSHOTS 1000ml (12 pack)] |
| **382** | 65 | 0.003208 | [FOUR ROSES SINGLE BARREL 750ml (6 pack)] |
| **383** | 627 | 0.003205 | [SEAGRAMS VO CANADIAN WHISKEY PET 1750ml (6 pa... |
| **384** | 121 | 0.003205 | [JIM BEAM HONEY MINI 50ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **385** | 505 | 0.003202 | [MARGARITAVILLE SILVER TEQUILA 750ml (12 pack)] |
| **386** | 341 | 0.003198 | [ON THE ROCKS COCKTAILS KNOB CREEK BOURBON OLD... |
| **387** | 663 | 0.003198 | [TEN HIGH 1750ml (6 pack), FIREBALL CINNAMON W... |
| **388** | 127 | 0.003195 | [PRAIRIE FIRE 750ml (12 pack)] |
| **389** | 606 | 0.003191 | [CROWN ROYAL 1750ml (6 pack), CROWN ROYAL 750m... |
| **390** | 51 | 0.003191 | [OLD FORESTER SIGNATURE 100PRF BOURBON 750ml (... |
| **391** | 318 | 0.003188 | [PARAMOUNT BLACKBERRY BRANDY 750ml (12 pack)] |
| **392** | 519 | 0.003181 | [CASAMIGOS REPOSADO 750ml (6 pack)] |
| **393** | 305 | 0.003171 | [GLENLIVET 12YR 750ml (12 pack)] |
| **394** | 360 | 0.003167 | [GRAND MARNIER CORDON ROUGE 750ml (12 pack)] |
| **395** | 314 | 0.003164 | [PAUL MASSON GRANDE AMBER BRANDY VS 750ml (12 ... |
| **396** | 157 | 0.003160 | [ABSOLUT SWEDISH VODKA 80PRF 375ml (24 pack)] |
| **397** | 706 | 0.003160 | [HAWKEYE VODKA 750ml (12 pack), HAWKEYE VODKA ... |
| **398** | 131 | 0.003160 | [JIM BEAM VANILLA 750ml (12 pack)] |
| **399** | 267 | 0.003153 | [BACARDI SUPERIOR 375ml (24 pack)] |
| **400** | 528 | 0.003153 | [JUAREZ GOLD 1000ml (12 pack)] |
| **401** | 747 | 0.003150 | [RUMPLE MINZE PEPPERMINT SCHNAPPS LIQUEUR 1000... |
| **402** | 152 | 0.003147 | [SEAGRAMS EXTRA DRY GIN 1750ml (6 pack)] |
| **403** | 650 | 0.003129 | [BLACK VELVET APPLE 1750ml (6 pack), BLACK VEL... |
| **404** | 488 | 0.003126 | [OLE SMOKY PEACH WHISKEY 750ml (6 pack)] |
| **405** | 181 | 0.003116 | [BARTON VODKA PET 750ml (12 pack)] |
| **406** | 224 | 0.003116 | [SMIRNOFF 80PRF 200ml (48 pack)] |
| **407** | 335 | 0.003112 | [JOSE CUERVO AUTHENTIC LIGHT WHITE PEACH MARGA... |
| **408** | 123 | 0.003105 | [BIRD DOG PEACH 750ml (6 pack)] |
| **409** | 711 | 0.003098 | [PARAMOUNT PEPPERMINT SCHNAPPS 1750ml (6 pack)... |
| **410** | 842 | 0.003095 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **411** | 444 | 0.003095 | [[99 PEPPERMINT MINI 50ml (1 pack), 99 PEPPERM... |
| **412** | 389 | 0.003095 | [DR MCGILLICUDDYS MENTHOLMINT 750ml (12 pack)] |
| **413** | 483 | 0.003092 | [SOUTHERN COMFORT 375ml (24 pack)] |
| **414** | 715 | 0.003085 | [BARTON NATURALS VODKA 1750ml (6 pack), FIREBA... |
| **415** | 478 | 0.003085 | [OLE SMOKY APPLE PIE MOONSHINE 70PRF 750ml (6 ... |
| **416** | 704 | 0.003085 | [HAWKEYE VODKA PET 750ml (12 pack), PARAMOUNT ... |
| **417** | 738 | 0.003078 | [TITOS HANDMADE VODKA MINI 50ml (5 pack), TITO... |
| **418** | 778 | 0.003074 | [FIREBALL CINNAMON WHISKEY 200ml (48 pack), FI... |
| **419** | 427 | 0.003074 | [SMIRNOFF WHIPPED CREAM 750ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **420** | 402 | 0.003067 | [KINKY BLUE 750ml (6 pack)] |
| **421** | 319 | 0.003061 | [PARAMOUNT BLACKBERRY BRANDY 1750ml (6 pack)] |
| **422** | 642 | 0.003054 | [BLACK VELVET 1750ml (6 pack), ELIJAH CRAIG 75... |
| **423** | 78 | 0.003050 | [JIM BEAM PET 750ml (12 pack)] |
| **424** | 724 | 0.003030 | [SKOL VODKA 1750ml (6 pack), FIREBALL CINNAMON... |
| **425** | 246 | 0.003030 | [DEEP EDDY RUBY RED GRAPEFRUIT 750ml (12 pack)] |
| **426** | 49 | 0.003023 | [EVAN WILLIAMS BOTTLED IN BOND 750ml (12 pack)] |
| **427** | 771 | 0.003019 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **428** | 9 | 0.003019 | [CROWN ROYAL BLACKBERRY 750ml (12 pack)] |
| **429** | 184 | 0.003009 | [BURNETTS VODKA 80PRF 1750ml (6 pack)] |
| **430** | 99 | 0.003006 | [FIVE STAR 375ml (24 pack)] |
| **431** | 723 | 0.003006 | [SKOL VODKA 1750ml (6 pack), FIREBALL CINNAMON... |
| **432** | 325 | 0.003002 | [CHI-CHI'S PINA COLADA 1750ml (6 pack)] |
| **433** | 361 | 0.002992 | [IL TRAMONTO LIMONCELLO 750ml (6 pack)] |
| **434** | 645 | 0.002988 | [DEEP EDDY LEMON 750ml (12 pack), BLACK VELVET... |
| **435** | 375 | 0.002988 | [MIDORI MELON LIQUEUR 750ml (12 pack)] |
| **436** | 292 | 0.002978 | [LADY BLIGH SPICED RUM 1750ml (6 pack)] |
| **437** | 641 | 0.002975 | [ADMIRAL NELSON SPICED 1750ml (6 pack), BLACK ... |
| **438** | 434 | 0.002964 | [IOWISH CREAM LIQUEUR 750ml (6 pack)] |
| **439** | 285 | 0.002961 | [CROSS KEYS RUM 750ml (12 pack)] |
| **440** | 23 | 0.002951 | [CROWN ROYAL 200ml (44 pack)] |
| **441** | 727 | 0.002947 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **442** | 166 | 0.002940 | [ABSOLUT VANILIA 750ml (12 pack)] |
| **443** | 217 | 0.002937 | [POPOV 80PRF 1750ml (6 pack)] |
| **444** | 614 | 0.002930 | [CAPTAIN MORGAN ORIGINAL SPICED 1000ml (12 pac... |
| **445** | 164 | 0.002930 | [ABSOLUT PEARS 750ml (12 pack)] |
| **446** | 744 | 0.002927 | [MALIBU COCONUT 1750ml (6 pack), MALIBU COCONU... |
| **447** | 623 | 0.002923 | [CAPTAIN MORGAN ORIGINAL SPICED 1000ml (12 pac... |
| **448** | 755 | 0.002916 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **449** | 807 | 0.002909 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **450** | 418 | 0.002906 | [SMIRNOFF BLUEBERRY 750ml (12 pack)] |
| **451** | 347 | 0.002892 | [CAMPARI ITALIAN APERITIVO 750ml (12 pack)] |
| **452** | 43 | 0.002892 | [JAMESON 1750ml (6 pack)] |
| **453** | 796 | 0.002892 | [DR MCGILLICUDDYS CHERRY 1750ml (6 pack), FIRE... |
| **454** | 413 | 0.002882 | [PARAMOUNT AMARETTO 1750ml (6 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **455** | 583 | 0.002878 | [CROWN ROYAL PEACH 750ml (12 pack), CAPTAIN MO... |
| **456** | 675 | 0.002871 | [CAPTAIN MORGAN ORIGINAL SPICED 750ml (12 pack... |
| **457** | 457 | 0.002858 | [DEKUYPER WATERMELON PUCKER 1000ml (12 pack)] |
| **458** | 732 | 0.002854 | [TITOS HANDMADE VODKA 1750ml (6 pack), TITOS H... |
| **459** | 659 | 0.002854 | [EVAN WILLIAMS BLACK 1750ml (6 pack), ADMIRAL ... |
| **460** | 216 | 0.002854 | [PHILLIPS VODKA 750ml (12 pack)] |
| **461** | 734 | 0.002847 | [TITOS HANDMADE VODKA MINI 50ml (5 pack), TITO... |
| **462** | 805 | 0.002847 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **463** | 381 | 0.002844 | [BAILEYS ORIGINAL IRISH CREAM 1000ml (12 pack)] |
| **464** | 545 | 0.002841 | [FIREBALL CINNAMON WHISKEY PET 750ml (12 pack)... |
| **465** | 188 | 0.002834 | [FIVE O'CLOCK VODKA 1000ml (12 pack)] |
| **466** | 251 | 0.002834 | [UV BLUE RASPBERRY 750ml (12 pack)] |
| **467** | 97 | 0.002820 | [BEAMS 8 STAR BL WHISKEY 1750ml (6 pack)] |
| **468** | 716 | 0.002820 | [BARTON NATURALS VODKA 1750ml (6 pack), FIREBA... |
| **469** | 763 | 0.002816 | [JOSE CUERVO ESPECIAL SILVER 750ml (12 pack), ... |
| **470** | 843 | 0.002816 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **471** | 720 | 0.002806 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **472** | 35 | 0.002806 | [RICH & RARE 1750ml (6 pack)] |
| **473** | 40 | 0.002803 | [JAMESON MINI 50ml (12 pack)] |
| **474** | 789 | 0.002799 | [FIREBALL CINNAMON WHISKEY 1000ml (12 pack), F... |
| **475** | 760 | 0.002799 | [JOSE CUERVO AUTHENTIC STRAWBERRY LIME MARGARI... |
| **476** | 133 | 0.002796 | [BOMBAY SAPPHIRE GIN 1000ml (12 pack)] |
| **477** | 683 | 0.002796 | [ABSOLUT SWEDISH VODKA 80PRF 750ml (12 pack), ... |
| **478** | 36 | 0.002789 | [CANADIAN RESERVE WHISKY 1750ml (6 pack)] |
| **479** | 401 | 0.002789 | [KETEL ONE BOTANICAL PEACH & ORANGE BLOSSOM 75... |
| **480** | 120 | 0.002779 | [TEMPLETON RYE 6YR 750ml (6 pack)] |
| **481** | 671 | 0.002768 | [SMIRNOFF 80PRF PET 1750ml (6 pack), SEAGRAMS ... |
| **482** | 518 | 0.002761 | [DON JULIO REPOSADO 750ml (6 pack)] |
| **483** | 48 | 0.002758 | [TULLAMORE DEW 750ml (12 pack)] |
| **484** | 787 | 0.002758 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **485** | 247 | 0.002758 | [DEEP EDDY SWEET TEA 750ml (12 pack)] |
| **486** | 575 | 0.002748 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **487** | 54 | 0.002748 | [BLADE & BOW 750ml (6 pack)] |
| **488** | 5 | 0.002748 | [CANADIAN CLUB WHISKY 1000ml (12 pack)] |
| **489** | 362 | 0.002741 | [TEQUILA ROSE LIQUEUR 750ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **490** | 613 | 0.002741 | [CAPTAIN MORGAN ORIGINAL SPICED 750ml (12 pack... |
| **491** | 383 | 0.002737 | [DR MCGILLICUDDYS APPLE PIE 750ml (12 pack)] |
| **492** | 302 | 0.002737 | [DEWARS WHITE LABEL SCOTCH 750ml (12 pack)] |
| **493** | 599 | 0.002727 | [CAPTAIN MORGAN ORIGINAL SPICED 750ml (12 pack... |
| **494** | 396 | 0.002720 | [RUMCHATA 1000ml (6 pack)] |
| **495** | 411 | 0.002717 | [PARAMOUNT AMARETTO 750ml (12 pack)] |
| **496** | 455 | 0.002717 | [DEKUYPER LUSCIOUS PEACHTREE PET 1750ml (6 pack)] |
| **497** | 737 | 0.002710 | [TITOS HANDMADE VODKA 200ml (24 pack), TITOS H... |
| **498** | 542 | 0.002706 | [FIREBALL CINNAMON WHISKEY PARTY BUCKET 50ml (... |
| **499** | 458 | 0.002706 | [ICE HOLE BUTTERSCOTCH SCHNAPPS 750ml (12 pack)] |
| **500** | 810 | 0.002700 | [RYANS CREAM LIQUEUR 1750ml (6 pack), FIREBALL... |
| **501** | 537 | 0.002693 | [BARTON VODKA 1750ml (6 pack), FIREBALL CINNAM... |
| **502** | 819 | 0.002693 | [ADMIRAL NELSON SPICED 1750ml (6 pack), BLACK ... |
| **503** | 673 | 0.002686 | [CAPTAIN MORGAN ORIGINAL SPICED MINI 50ml (12 ... |
| **504** | 547 | 0.002682 | [FIREBALL CINNAMON WHISKEY PET 1750ml (6 pack)... |
| **505** | 713 | 0.002682 | [BLUE OX VODKA 750ml (12 pack), BLUE OX VODKA ... |
| **506** | 312 | 0.002682 | [E & J VS 1750ml (6 pack)] |
| **507** | 489 | 0.002682 | [SKREWBALL PEANUT BUTTER WHISKEY MINI 50ml (12... |
| **508** | 845 | 0.002679 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **509** | 536 | 0.002679 | [FRIS DANISH VODKA 1750ml (6 pack), FIREBALL C... |
| **510** | 662 | 0.002675 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **511** | 580 | 0.002675 | [CROWN ROYAL PEACH 750ml (12 pack), SMIRNOFF 8... |
| **512** | 155 | 0.002672 | [ABSOLUT LIME 750ml (12 pack)] |
| **513** | 414 | 0.002672 | [[THE ORIGINAL PICKLE SHOT DILL PICKLE VODKA M... |
| **514** | 631 | 0.002669 | [BLACK VELVET 200ml (48 pack), BLACK VELVET 17... |
| **515** | 657 | 0.002669 | [BULLEIT BOURBON 750ml (12 pack), SEAGRAMS 7 C... |
| **516** | 320 | 0.002651 | [MONKEY SHOULDER 750ml (6 pack)] |
| **517** | 179 | 0.002638 | [REYKA VODKA 750ml (6 pack)] |
| **518** | 672 | 0.002631 | [SEAGRAMS 7 CROWN 1750ml (6 pack), CAPTAIN MOR... |
| **519** | 748 | 0.002627 | [PARAMOUNT GOLD RUM 1750ml (6 pack), PARAMOUNT... |
| **520** | 431 | 0.002624 | [PARAMOUNT CREME DE CACAO WHITE 750ml (12 pack)] |
| **521** | 741 | 0.002624 | [TITOS HANDMADE VODKA MINI 50ml (5 pack), TITO... |
| **522** | 582 | 0.002620 | [CROWN ROYAL PEACH 750ml (12 pack), CAPTAIN MO... |
| **523** | 382 | 0.002617 | [DR MCGILLICUDDYS APPLE PIE MINI 50ml (12 pack)] |
| **524** | 185 | 0.002614 | [PINNACLE WHIPPED 750ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **525** | 212 | 0.002603 | [BLUE OX VODKA 375ml (24 pack)] |
| **526** | 337 | 0.002603 | [DESERT ISLAND LONG ISLAND ICED TEA COCKTAIL 1... |
| **527** | 172 | 0.002603 | [GREY GOOSE 1750ml (6 pack)] |
| **528** | 456 | 0.002600 | [DEKUYPER WATERMELON PUCKER 750ml (12 pack)] |
| **529** | 143 | 0.002600 | [BURNETTS GIN LONDON DRY 1750ml (6 pack)] |
| **530** | 506 | 0.002600 | [MILAGRO SILVER 750ml (6 pack)] |
| **531** | 553 | 0.002596 | [BLACK VELVET TOASTED CARAMEL 1750ml (6 pack),... |
| **532** | 577 | 0.002596 | [CROWN ROYAL PEACH 750ml (12 pack), BULLEIT BO... |
| **533** | 428 | 0.002596 | [SMIRNOFF RED, WHITE & BERRY 750ml (12 pack)] |
| **534** | 405 | 0.002583 | [KINKY BLUE MINI 50ml (6 pack)] |
| **535** | 398 | 0.002583 | [CAPTAIN MORGAN CHERRY VANILLA 750ml (12 pack)] |
| **536** | 104 | 0.002583 | [SEAGRAMS 7 CROWN 750ml (12 pack)] |
| **537** | 424 | 0.002576 | [SAINTS N SINNERS APPLE PIE 750ml (12 pack)] |
| **538** | 766 | 0.002576 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **539** | 446 | 0.002576 | [PARAMOUNT PEPPERMINT SCHNAPPS 1000ml (12 pack)] |
| **540** | 494 | 0.002572 | [JOSE CUERVO ESPECIAL SILVER 375ml (24 pack)] |
| **541** | 72 | 0.002572 | [JIM BEAM 200ml (48 pack)] |
| **542** | 272 | 0.002569 | [BACARDI LIMON 750ml (12 pack)] |
| **543** | 556 | 0.002569 | [BLACK VELVET TOASTED CARAMEL 1750ml (6 pack),... |
| **544** | 514 | 0.002565 | [SAUZA BLANCO SILVER 750ml (12 pack)] |
| **545** | 616 | 0.002562 | [CROWN ROYAL 1000ml (12 pack), SEAGRAMS 7 CROW... |
| **546** | 799 | 0.002562 | [RYANS CREAM LIQUEUR 1750ml (6 pack), FIREBALL... |
| **547** | 818 | 0.002559 | [FIREBALL CINNAMON WHISKEY 100ML CARRIER 100ml... |
| **548** | 719 | 0.002559 | [NIKOLAI VODKA 1750ml (6 pack), FIREBALL CINNA... |
| **549** | 598 | 0.002559 | [CAPTAIN MORGAN ORIGINAL SPICED MINI 50ml (12 ... |
| **550** | 532 | 0.002555 | [SCORESBY RARE SCOTCH 1750ml (6 pack), FIREBAL... |
| **551** | 785 | 0.002555 | [FIREBALL CINNAMON WHISKEY 750ml (12 pack), FI... |
| **552** | 740 | 0.002548 | [TITOS HANDMADE VODKA 200ml (24 pack), TITOS H... |
| **553** | 154 | 0.002548 | [ABSOLUT WILD BERRY 750ml (6 pack)] |
| **554** | 109 | 0.002545 | [GENTLEMAN JACK 750ml (12 pack)] |
| **555** | 541 | 0.002545 | [PLATINUM 7X VODKA 1750ml (6 pack), FIREBALL C... |
| **556** | 313 | 0.002541 | [PAUL MASSON GRANDE AMBER BRANDY VS 375ml (24 ... |
| **557** | 118 | 0.002541 | [SAZERAC RYE 750ml (6 pack)] |
| **558** | 560 | 0.002538 | [BLACK VELVET TOASTED CARAMEL 1750ml (6 pack),... |
| **559** | 46 | 0.002538 | [PROPER NO. TWELVE 750ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **560** | 468 | 0.002538 | [REVELTON WHISKEY AND CREAM LIQUEUR 750ml (6 p... |
| **561** | 634 | 0.002531 | [BLACK VELVET 375ml (24 pack), ADMIRAL NELSON ... |
| **562** | 280 | 0.002531 | [CAPTAIN MORGAN ORIGINAL SPICED PET 200ml (48 ... |
| **563** | 309 | 0.002514 | [E & J VSOP 750ml (12 pack)] |
| **564** | 603 | 0.002514 | [CROWN ROYAL 1750ml (6 pack), CROWN ROYAL REGA... |
| **565** | 445 | 0.002514 | [PARAMOUNT PEPPERMINT SCHNAPPS PET 750ml (12 p... |
| **566** | 712 | 0.002510 | [JUAREZ TRIPLE SEC 1000ml (12 pack), HAWKEYE V... |
| **567** | 502 | 0.002500 | [FAMILIA CAMARENA SILVER 750ml (12 pack)] |
| **568** | 126 | 0.002500 | [TIN CUP 750ml (12 pack)] |
| **569** | 660 | 0.002500 | [CAPTAIN MORGAN ORIGINAL SPICED BARREL 1750ml ... |
| **570** | 50 | 0.002490 | [EVAN WILLIAMS WHITE 1750ml (6 pack)] |
| **571** | 463 | 0.002483 | [[99 STRAWBERRIES MINI 50ml (1 pack), 99 STRAW... |
| **572** | 199 | 0.002483 | [BLUE OX VODKA 1000ml (12 pack)] |
| **573** | 700 | 0.002479 | [HAWKEYE VODKA PET 750ml (12 pack), HAWKEYE VO... |
| **574** | 830 | 0.002476 | [BARTON VODKA 1750ml (6 pack), FIREBALL CINNAM... |
| **575** | 593 | 0.002473 | [BULLEIT BOURBON 750ml (12 pack), CROWN ROYAL ... |
| **576** | 85 | 0.002473 | [MAKERS 46 750ml (6 pack)] |
| **577** | 479 | 0.002473 | [SOUTHERN HOST WHISKEY LIQUEUR 750ml (12 pack)] |
| **578** | 210 | 0.002469 | [NEW AMSTERDAM 80PRF 1750ml (6 pack)] |
| **579** | 476 | 0.002469 | [JACK DANIELS TENNESSEE FIRE 750ml (12 pack)] |
| **580** | 666 | 0.002459 | [SEAGRAMS 7 CROWN 1000ml (12 pack), CAPTAIN MO... |
| **581** | 317 | 0.002459 | [LAPHROAIG 10YR 750ml (12 pack)] |
| **582** | 804 | 0.002459 | [DR MCGILLICUDDYS CHERRY MINI 50ml (12 pack), ... |
| **583** | 421 | 0.002448 | [TRAVIS HASSE COW PIE 750ml (6 pack)] |
| **584** | 449 | 0.002448 | [DEKUYPER SOUR APPLE PUCKER 1000ml (12 pack)] |
| **585** | 612 | 0.002448 | [CAPTAIN MORGAN ORIGINAL SPICED MINI 50ml (12 ... |
| **586** | 214 | 0.002448 | [PHILLIPS VODKA 1000ml (12 pack)] |
| **587** | 567 | 0.002438 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **588** | 493 | 0.002438 | [CORAZON DE AGAVE BLANCO 750ml (6 pack)] |
| **589** | 331 | 0.002438 | [JOSE CUERVO AUTHENTIC MANGO MARGARITA 1750ml ... |
| **590** | 275 | 0.002438 | [CAPTAIN MORGAN 100PRF SPICED RUM 750ml (12 pa... |
| **591** | 578 | 0.002435 | [CROWN ROYAL PEACH 750ml (12 pack), CAPTAIN MO... |
| **592** | 564 | 0.002435 | [ADMIRAL NELSON SPICED 1750ml (6 pack), BLACK ... |
| **593** | 803 | 0.002435 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **594** | 186 | 0.002431 | [PEARL CUCUMBER 750ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **595** | 827 | 0.002424 | [FRIS DANISH VODKA 1750ml (6 pack), FIREBALL C... |
| **596** | 187 | 0.002421 | [FIVE O'CLOCK VODKA 375ml (24 pack)] |
| **597** | 243 | 0.002418 | [WESTERN SON BLUEBERRY 750ml (12 pack)] |
| **598** | 834 | 0.002418 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **599** | 308 | 0.002414 | [E & J XO 750ml (12 pack)] |
| **600** | 596 | 0.002411 | [SMIRNOFF 80PRF PET 1750ml (6 pack), CROWN ROY... |
| **601** | 227 | 0.002407 | [SMIRNOFF 80PRF 1000ml (12 pack)] |
| **602** | 832 | 0.002407 | [TITOS HANDMADE VODKA 1750ml (6 pack), TITOS H... |
| **603** | 550 | 0.002407 | [FIREBALL CINNAMON WHISKEY 100ML CARRIER 100ml... |
| **604** | 698 | 0.002400 | [FLEISCHMANNS 80PRF VODKA 1750ml (6 pack), FIR... |
| **605** | 795 | 0.002397 | [DR MCGILLICUDDYS CHERRY 750ml (12 pack), FIRE... |
| **606** | 844 | 0.002397 | [FIREBALL CINNAMON WHISKEY PET 1750ml (6 pack)... |
| **607** | 629 | 0.002393 | [BLACK VELVET MINI 50ml (8 pack), BLACK VELVET... |
| **608** | 823 | 0.002393 | [CROWN ROYAL 1000ml (12 pack), CAPTAIN MORGAN ... |
| **609** | 436 | 0.002393 | [OLE SMOKY ORANGE SHINESICLE MOONSHINE 750ml (... |
| **610** | 680 | 0.002393 | [TANQUERAY GIN 1000ml (12 pack), CAPTAIN MORGA... |
| **611** | 544 | 0.002390 | [FIREBALL CINNAMON WHISKEY PARTY BUCKET 50ml (... |
| **612** | 558 | 0.002387 | [EVAN WILLIAMS BLACK 1750ml (6 pack), BLACK VE... |
| **613** | 618 | 0.002380 | [RUMPLE MINZE PEPPERMINT SCHNAPPS LIQUEUR 1000... |
| **614** | 89 | 0.002380 | [[1792 SMALL BATCH 750ml (6 pack), RIDGEMONT R... |
| **615** | 327 | 0.002376 | [CAPTAIN MORGAN LONG ISLAND ICED TEA 1750ml (6... |
| **616** | 400 | 0.002376 | [KETEL ONE BOTANICAL CUCUMBER & MINT 750ml (12... |
| **617** | 252 | 0.002376 | [UV BLUE RASPBERRY 1000ml (12 pack)] |
| **618** | 591 | 0.002373 | [CROWN ROYAL 1000ml (12 pack), CROWN ROYAL REG... |
| **619** | 202 | 0.002369 | [MCCORMICK 80PRF VODKA 1000ml (12 pack)] |
| **620** | 764 | 0.002366 | [JOSE CUERVO ESPECIAL REPOSADO 750ml (12 pack)... |
| **621** | 691 | 0.002366 | [BARTON VODKA 1000ml (12 pack), FIREBALL CINNA... |
| **622** | 450 | 0.002363 | [DEKUYPER BUTTERSHOTS 750ml (12 pack)] |
| **623** | 588 | 0.002349 | [CAPTAIN MORGAN ORIGINAL SPICED MINI 50ml (12 ... |
| **624** | 527 | 0.002345 | [CORRALEJO REPOSADO 750ml (6 pack)] |
| **625** | 443 | 0.002342 | [ARROW PEPPERMINT SCHNAPPS 1750ml (6 pack)] |
| **626** | 339 | 0.002342 | [ON THE ROCKS COCKTAILS EFFEN COSMOPOLITAN 375... |
| **627** | 562 | 0.002342 | [BLACK VELVET 1000ml (12 pack), BLACK VELVET T... |
| **628** | 776 | 0.002338 | [FIREBALL CINNAMON WHISKEY 200ml (48 pack), FI... |
| **629** | 55 | 0.002338 | [BLANTONS BOURBON 750ml (6 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **630** | 648 | 0.002335 | [CHRISTIAN BROS BRANDY 750ml (12 pack), BLACK ... |
| **631** | 652 | 0.002335 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **632** | 753 | 0.002335 | [CHI-CHI'S MEXICAN MUDSLIDE 1750ml (6 pack), F... |
| **633** | 808 | 0.002335 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **634** | 66 | 0.002328 | [FOUR ROSES BOURBON 1750ml (6 pack)] |
| **635** | 342 | 0.002328 | [SALVADORS ORIGINAL MARGARITA PET 1750ml (6 pa... |
| **636** | 568 | 0.002328 | [CROWN ROYAL REGAL APPLE 750ml (12 pack), CROW... |
| **637** | 707 | 0.002325 | [HAWKEYE VODKA 1000ml (12 pack), HAWKEYE VODKA... |
| **638** | 752 | 0.002321 | [CHI-CHI'S MEXICAN MUDSLIDE 1750ml (6 pack), F... |
| **639** | 429 | 0.002321 | [SMIRNOFF ZERO SUGAR INFUSION CUCUMBER & LIME ... |
| **640** | 533 | 0.002321 | [FIREBALL CINNAMON WHISKEY 100ML CARRIER 100ml... |
| **641** | 471 | 0.002321 | [BLACK VELVET APPLE 750ml (12 pack)] |
| **642** | 128 | 0.002321 | [JIM BEAM KENTUCKY FIRE 750ml (12 pack)] |
| **643** | 625 | 0.002321 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **644** | 358 | 0.002318 | [FRANGELICO LIQUEUR 750ml (12 pack)] |
| **645** | 658 | 0.002314 | [BULLEIT BOURBON 750ml (12 pack), CAPTAIN MORG... |
| **646** | 504 | 0.002314 | [LUNAZUL BLANCO 750ml (12 pack)] |
| **647** | 93 | 0.002314 | [WILD TURKEY 81 750ml (12 pack)] |
| **648** | 330 | 0.002311 | [JOSE CUERVO AUTHENTIC DOUBLE STRENGTH MARGARI... |
| **649** | 45 | 0.002311 | [JAMESON BLACK BARREL 750ml (12 pack)] |
| **650** | 162 | 0.002311 | [ABSOLUT CITRON 1000ml (12 pack)] |
| **651** | 517 | 0.002307 | [CAZADORES REPOSADO 750ml (12 pack)] |
| **652** | 722 | 0.002307 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **653** | 630 | 0.002304 | [BLACK VELVET 375ml (24 pack), BLACK VELVET 20... |
| **654** | 794 | 0.002301 | [DR MCGILLICUDDYS CHERRY MINI 50ml (12 pack), ... |
| **655** | 788 | 0.002301 | [FIREBALL CINNAMON WHISKEY 1000ml (12 pack), F... |
| **656** | 497 | 0.002301 | [JOSE CUERVO ESPECIAL SILVER 1750ml (6 pack)] |
| **657** | 448 | 0.002297 | [DEKUYPER SOUR APPLE PUCKER 750ml (12 pack)] |
| **658** | 278 | 0.002297 | [CAPTAIN MORGAN WHITE 750ml (12 pack)] |
| **659** | 654 | 0.002297 | [JAMESON 750ml (12 pack), MALIBU COCONUT 750ml... |
| **660** | 535 | 0.002294 | [TEN HIGH 1750ml (6 pack), FIREBALL CINNAMON W... |
| **661** | 759 | 0.002290 | [CHI-CHI'S SKINNY MARGARITA 1750ml (6 pack), F... |
| **662** | 684 | 0.002290 | [BARTON VODKA 1750ml (6 pack), FRIS DANISH VOD... |
| **663** | 790 | 0.002290 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **664** | 161 | 0.002287 | [ABSOLUT APEACH 750ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **665** | 615 | 0.002287 | [CROWN ROYAL 1000ml (12 pack), CROWN ROYAL 175... |
| **666** | 132 | 0.002283 | [BEEFEATER GIN 750ml (12 pack)] |
| **667** | 460 | 0.002283 | [99 BANANAS 100ml (48 pack)] |
| **668** | 798 | 0.002277 | [BUFFALO TRACE BOURBON CREAM 750ml (12 pack), ... |
| **669** | 287 | 0.002277 | [RONRICO SILVER LABEL RUM 1750ml (6 pack)] |
| **670** | 594 | 0.002277 | [CAPTAIN MORGAN ORIGINAL SPICED BARREL 1750ml ... |
| **671** | 708 | 0.002273 | [EVERCLEAR 151 750ml (12 pack), HAWKEYE VODKA ... |
| **672** | 718 | 0.002273 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **673** | 168 | 0.002270 | [FRIS DANISH VODKA 750ml (12 pack)] |
| **674** | 151 | 0.002270 | [SEAGRAMS EXTRA DRY GIN 750ml (12 pack)] |
| **675** | 703 | 0.002270 | [HAWKEYE VODKA PET 750ml (12 pack), PARAMOUNT ... |
| **676** | 141 | 0.002266 | [TANQUERAY RANGPUR GIN 750ml (12 pack)] |
| **677** | 379 | 0.002266 | [KAPALI COFFEE LIQUEUR 750ml (12 pack)] |
| **678** | 837 | 0.002266 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **679** | 91 | 0.002263 | [BENCHMARK OLD NO 8 750ml (12 pack)] |
| **680** | 565 | 0.002263 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **681** | 254 | 0.002259 | [UV CAKE 750ml (12 pack)] |
| **682** | 814 | 0.002259 | [JOSE CUERVO ESPECIAL REPOSADO 750ml (12 pack)... |
| **683** | 288 | 0.002256 | [MALIBU STRAWBERRY 750ml (12 pack)] |
| **684** | 529 | 0.002252 | [MARGARITAVILLE GOLD TEQUILA 750ml (12 pack)] |
| **685** | 321 | 0.002252 | [CHI-CHI'S LONG ISLAND ICED TEA 1750ml (6 pack)] |
| **686** | 835 | 0.002249 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **687** | 725 | 0.002249 | [CAPTAIN MORGAN ORIGINAL SPICED MINI 50ml (12 ... |
| **688** | 610 | 0.002249 | [SMIRNOFF 80PRF PET 1750ml (6 pack), CROWN ROY... |
| **689** | 836 | 0.002246 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **690** | 419 | 0.002246 | [SMIRNOFF CHERRY 750ml (12 pack)] |
| **691** | 800 | 0.002242 | [SOUTHERN COMFORT MINI 50ml (10 pack), FIREBAL... |
| **692** | 171 | 0.002235 | [GREY GOOSE 375ml (12 pack)] |
| **693** | 762 | 0.002235 | [JOSE CUERVO AUTHENTIC LIME MARGARITA 1750ml (... |
| **694** | 646 | 0.002235 | [ADMIRAL NELSON SPICED PET 750ml (12 pack), BL... |
| **695** | 475 | 0.002232 | [JACK DANIELS TENNESSEE FIRE MINI 50ml (12 pack)] |
| **696** | 70 | 0.002232 | [JIM BEAM BLACK 750ml (12 pack)] |
| **697** | 681 | 0.002228 | [PARAMOUNT GIN 1750ml (6 pack), HAWKEYE VODKA ... |
| **698** | 581 | 0.002228 | [CROWN ROYAL PEACH 750ml (12 pack), CAPTAIN MO... |
| **699** | 376 | 0.002228 | [KAHLUA COFFEE MINI 50ml (12 pack)] |

| | index | support | itemsets |
|---|---|---|---|
| **700** | 34 | 0.002228 | [CANADIAN MIST WHISKY PET 1750ml (6 pack)] |
| **701** | 364 | 0.002225 | [JAGERMEISTER LIQUEUR 375ml (24 pack)] |
| **702** | 597 | 0.002225 | [CROWN ROYAL REGAL APPLE 750ml (12 pack), CAPT... |
| **703** | 90 | 0.002225 | [BENCHMARK FULL PROOF 750ml (12 pack)] |
| **704** | 245 | 0.002222 | [NEW AMSTERDAM PINEAPPLE 750ml (12 pack)] |
| **705** | 640 | 0.002222 | [EVAN WILLIAMS BLACK 1750ml (6 pack), BLACK VE... |
| **706** | 833 | 0.002218 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **707** | 146 | 0.002218 | [GORDONS GIN LONDON DRY 750ml (12 pack)] |
| **708** | 566 | 0.002218 | [CROWN ROYAL 1750ml (6 pack), CROWN ROYAL PEAC... |
| **709** | 107 | 0.002215 | [SEAGRAMS 7 CROWN PET FLASK 750ml (12 pack)] |
| **710** | 304 | 0.002211 | [GLENFIDDICH 12YR SPECIAL RESERVE 750ml (12 pa... |
| **711** | 540 | 0.002211 | [SKOL VODKA 1750ml (6 pack), FIREBALL CINNAMON... |
| **712** | 496 | 0.002211 | [JOSE CUERVO ESPECIAL SILVER 1000ml (12 pack)] |
| **713** | 607 | 0.002211 | [BULLEIT BOURBON 750ml (12 pack), CROWN ROYAL ... |
| **714** | 751 | 0.002211 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **715** | 98 | 0.002208 | [FIVE STAR PET 750ml (12 pack)] |
| **716** | 638 | 0.002201 | [ADMIRAL NELSON SPICED 1750ml (6 pack), BLACK ... |
| **717** | 632 | 0.002197 | [BLACK VELVET 375ml (24 pack), BLACK VELVET PE... |
| **718** | 7 | 0.002197 | [CROWN ROYAL PEACH 1000ml (12 pack)] |
| **719** | 310 | 0.002191 | [E & J VS 375ml (24 pack)] |
| **720** | 290 | 0.002187 | [CRUZAN COCONUT 750ml (12 pack)] |
| **721** | 774 | 0.002187 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **722** | 809 | 0.002184 | [BUFFALO TRACE BOURBON CREAM 750ml (12 pack), ... |
| **723** | 841 | 0.002184 | [FIREBALL CINNAMON WHISKEY 200ml (48 pack), FI... |
| **724** | 667 | 0.002184 | [SEAGRAMS 7 CROWN 1750ml (6 pack), KETEL ONE 7... |
| **725** | 831 | 0.002184 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **726** | 554 | 0.002180 | [BLACK VELVET 375ml (24 pack), BLACK VELVET TO... |
| **727** | 608 | 0.002180 | [CAPTAIN MORGAN ORIGINAL SPICED BARREL 1750ml ... |
| **728** | 746 | 0.002180 | [CAPTAIN MORGAN ORIGINAL SPICED 750ml (12 pack... |
| **729** | 749 | 0.002173 | [CHI-CHI'S MARGARITA 1750ml (6 pack), CHI-CHI'... |
| **730** | 298 | 0.002173 | [KRAKEN BLACK SPICED RUM 1750ml (6 pack)] |
| **731** | 526 | 0.002170 | [JOSE CUERVO ESPECIAL REPOSADO SQUARE 375ml (1... |
| **732** | 781 | 0.002170 | [FIREBALL CINNAMON WHISKEY PET 750ml (12 pack)... |
| **733** | 665 | 0.002170 | [FIVE STAR 1750ml (6 pack), FIVE O'CLOCK VODKA... |
| **734** | 829 | 0.002170 | [BARTON VODKA 1750ml (6 pack), FIREBALL CINNAM... |

| | index | support | itemsets |
|---|---|---|---|
| **735** | 586 | 0.002170 | [CROWN ROYAL MINI 50ml (10 pack), CROWN ROYAL ... |
| **736** | 750 | 0.002167 | [CHI-CHI'S GOLD MARGARITA 1750ml (6 pack), FIR... |
| **737** | 689 | 0.002167 | [BACARDI SUPERIOR 1000ml (12 pack), GREY GOOSE... |
| **738** | 611 | 0.002167 | [CAPTAIN MORGAN ORIGINAL SPICED PET 750ml (12 ... |
| **739** | 452 | 0.002163 | [DEKUYPER STRAWBERRY PUCKER 750ml (12 pack)] |
| **740** | 100 | 0.002160 | [FIVE STAR 750ml (12 pack)] |
| **741** | 238 | 0.002160 | [WISCONSIN CLUB VODKA 1750ml (6 pack)] |
| **742** | 636 | 0.002160 | [BLACK VELVET 750ml (12 pack), ADMIRAL NELSON ... |
| **743** | 651 | 0.002156 | [CANADIAN LTD WHISKY 1750ml (6 pack), FIREBALL... |
| **744** | 534 | 0.002156 | [SEAGRAMS VO CANADIAN WHISKEY PET 1750ml (6 pa... |
| **745** | 758 | 0.002156 | [CHI-CHI'S SKINNY MARGARITA 1750ml (6 pack), F... |
| **746** | 430 | 0.002156 | [PARAMOUNT CREME DE CACAO DARK 750ml (12 pack)] |
| **747** | 531 | 0.002156 | [SCORESBY RARE SCOTCH 1750ml (6 pack), FIREBAL... |
| **748** | 806 | 0.002153 | [FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (1... |
| **749** | 459 | 0.002153 | [99 BANANAS 750ml (12 pack)] |
| **750** | 394 | 0.002149 | [CAPTAIN MORGAN SLICED APPLE 750ml (12 pack)] |
| **751** | 769 | 0.002149 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **752** | 685 | 0.002149 | [FRIS DANISH VODKA 1750ml (6 pack), PLATINUM 7... |
| **753** | 670 | 0.002146 | [SMIRNOFF 80PRF PET 750ml (12 pack), SEAGRAMS ... |
| **754** | 259 | 0.002146 | [ADMIRAL NELSON SPICED MINI 50ml (8 pack)] |
| **755** | 220 | 0.002139 | [SEAGRAMS EXTRA SMOOTH VODKA 750ml (12 pack)] |
| **756** | 561 | 0.002136 | [BLACK VELVET TOASTED CARAMEL 1750ml (6 pack),... |
| **757** | 826 | 0.002136 | [FRIS DANISH VODKA 1750ml (6 pack), FIREBALL C... |
| **758** | 622 | 0.002132 | [CAPTAIN MORGAN ORIGINAL SPICED 750ml (12 pack... |
| **759** | 178 | 0.002132 | [STOLICHNAYA 80PRF 750ml (12 pack)] |
| **760** | 604 | 0.002132 | [CROWN ROYAL 375ml (24 pack), SEAGRAMS 7 CROWN... |
| **761** | 286 | 0.002129 | [APPLETON ESTATE SIGNATURE BLEND 750ml (12 pack)] |
| **762** | 589 | 0.002129 | [CROWN ROYAL REGAL APPLE 375ml (24 pack), SEAG... |
| **763** | 175 | 0.002125 | [KETEL ONE 1000ml (12 pack)] |
| **764** | 825 | 0.002115 | [FRIS DANISH VODKA 1750ml (6 pack), FIREBALL C... |
| **765** | 693 | 0.002115 | [BARTON VODKA 1750ml (6 pack), CHI-CHI'S MARGA... |
| **766** | 340 | 0.002111 | [ON THE ROCKS COCKTAILS HORNITOS MARGARITA 375... |
| **767** | 824 | 0.002108 | [SEAGRAMS VO CANADIAN WHISKEY PET 1750ml (6 pa... |
| **768** | 669 | 0.002108 | [SMIRNOFF 80PRF 375ml (24 pack), SEAGRAMS 7 CR... |
| **769** | 801 | 0.002108 | [LAUDERS 1750ml (6 pack), FIREBALL CINNAMON WH... |

| | index | support | itemsets |
|---|---|---|---|
| **770** | 264 | 0.002105 | [BACARDI GOLD 1750ml (6 pack)] |
| **771** | 679 | 0.002101 | [JACK DANIELS TENNESSEE HONEY 750ml (12 pack),... |
| **772** | 812 | 0.002098 | [LAUDERS 1750ml (6 pack), FIREBALL CINNAMON WH... |
| **773** | 516 | 0.002094 | [TORTILLA WHITE 1000ml (12 pack)] |
| **774** | 81 | 0.002094 | [MAKERS MARK REPLICA 375ml (12 pack)] |
| **775** | 838 | 0.002094 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **776** | 303 | 0.002094 | [DEWARS WHITE LABEL SCOTCH 1750ml (6 pack)] |
| **777** | 384 | 0.002094 | [DR MCGILLICUDDYS BUTTERSCOTCH 750ml (12 pack)] |
| **778** | 699 | 0.002091 | [HAWKEYE VODKA MINI 50ml (12 pack), HAWKEYE VO... |
| **779** | 68 | 0.002091 | [LARCENY SMALL BATCH 750ml (12 pack)] |
| **780** | 656 | 0.002087 | [BUFFALO TRACE BOURBON 750ml (12 pack), BLANTO... |
| **781** | 743 | 0.002084 | [TITOS HANDMADE VODKA MINI 50ml (5 pack), TITO... |
| **782** | 390 | 0.002084 | [DR MCGILLICUDDYS MENTHOLMINT 1750ml (6 pack)] |
| **783** | 410 | 0.002084 | [PAMA POMEGRANATE LIQUEUR 750ml (6 pack)] |
| **784** | 813 | 0.002084 | [THE ORIGINAL PICKLE SHOT SPICY PICKLE VODKA 7... |
| **785** | 677 | 0.002081 | [BAILEYS ORIGINAL IRISH CREAM 750ml (12 pack),... |
| **786** | 821 | 0.002077 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **787** | 653 | 0.002077 | [ABSOLUT SWEDISH VODKA 80PRF 750ml (12 pack), ... |
| **788** | 828 | 0.002074 | [BARTON VODKA 1750ml (6 pack), FIREBALL CINNAM... |
| **789** | 668 | 0.002074 | [SMIRNOFF 80PRF MINI 50ml (12 pack), SEAGRAMS ... |
| **790** | 462 | 0.002074 | [99 PEPPERMINT 100ml (48 pack)] |
| **791** | 570 | 0.002074 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **792** | 500 | 0.002074 | [EL MAYOR BLANCO 750ml (6 pack)] |
| **793** | 291 | 0.002074 | [CRUZAN VANILLA 750ml (12 pack)] |
| **794** | 822 | 0.002070 | [CROWN ROYAL REGAL APPLE 750ml (12 pack), SEAG... |
| **795** | 407 | 0.002070 | [KINKY RED MINI 50ml (6 pack)] |
| **796** | 374 | 0.002070 | [21SEEDS CUCUMBER JALAPENO TEQUILA 750ml (6 pa... |
| **797** | 110 | 0.002070 | [SLIPKNOT IOWA WHISKEY NO. 9 750ml (6 pack)] |
| **798** | 491 | 0.002067 | [CABO WABO BLANCO 750ml (6 pack)] |
| **799** | 587 | 0.002067 | [CROWN ROYAL REGAL APPLE MINI 50ml (10 pack), ... |
| **800** | 811 | 0.002067 | [SOUTHERN COMFORT MINI 50ml (10 pack), FIREBAL... |
| **801** | 571 | 0.002067 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **802** | 682 | 0.002067 | [ABSOLUT SWEDISH VODKA 80PRF 750ml (12 pack), ... |
| **803** | 555 | 0.002060 | [BLACK VELVET TOASTED CARAMEL 1750ml (6 pack),... |
| **804** | 714 | 0.002060 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |

| | index | support | itemsets |
|---|---|---|---|
| **805** | 655 | 0.002060 | [JAMESON 1000ml (12 pack), MALIBU COCONUT 1000... |
| **806** | 69 | 0.002056 | [[JEFFERSONS BOURBON 750ml (6 pack), JEFFERSON... |
| **807** | 721 | 0.002056 | [ROW VODKA 750ml (12 pack), ROW VODKA 1750ml (... |
| **808** | 624 | 0.002053 | [FRIS DANISH VODKA 1750ml (6 pack), SEAGRAMS V... |
| **809** | 742 | 0.002053 | [TITOS HANDMADE VODKA MINI 50ml (5 pack), TITO... |
| **810** | 605 | 0.002050 | [CROWN ROYAL 1000ml (12 pack), CROWN ROYAL 750... |
| **811** | 786 | 0.002050 | [FIREBALL CINNAMON WHISKEY PET 1750ml (6 pack)... |
| **812** | 367 | 0.002046 | [JAGERMEISTER LIQUEUR 1750ml (6 pack)] |
| **813** | 406 | 0.002043 | [KINKY GREEN MINI 50ml (6 pack)] |
| **814** | 775 | 0.002043 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **815** | 745 | 0.002043 | [CAPTAIN MORGAN ORIGINAL SPICED MINI 50ml (12 ... |
| **816** | 432 | 0.002039 | [PARAMOUNT CREME DE MENTHE GREEN 750ml (12 pack)] |
| **817** | 674 | 0.002039 | [SEAGRAMS 7 CROWN 1750ml (6 pack), CAPTAIN MOR... |
| **818** | 697 | 0.002036 | [FLEISCHMANNS 80PRF VODKA 1750ml (6 pack), FIR... |
| **819** | 692 | 0.002036 | [BARTON VODKA 1750ml (6 pack), PLATINUM 7X VOD... |
| **820** | 47 | 0.002036 | [REDBREAST 12YR 750ml (6 pack)] |
| **821** | 620 | 0.002032 | [SMIRNOFF 80PRF PET 1750ml (6 pack), CROWN ROY... |
| **822** | 284 | 0.002032 | [BALVENIE 12YR DOUBLEWOOD 750ml (12 pack)] |
| **823** | 797 | 0.002029 | [DI AMORE AMARETTO 750ml (12 pack), FIREBALL C... |
| **824** | 239 | 0.002029 | [DEEP EDDY LIME FLAVORED VODKA 750ml (12 pack)] |
| **825** | 336 | 0.002029 | [JOSE CUERVO AUTHENTIC LIME LIGHT 200ml (6 pack)] |
| **826** | 573 | 0.002029 | [CROWN ROYAL PEACH 750ml (12 pack), CROWN ROYA... |
| **827** | 469 | 0.002029 | [OLE SMOKY MINT CHOCOLATE CHIP CREAM WHISKEY 7... |
| **828** | 690 | 0.002026 | [BARTON VODKA 1000ml (12 pack), FIREBALL CINNA... |
| **829** | 539 | 0.002026 | [NIKOLAI VODKA 1750ml (6 pack), FIREBALL CINNA... |
| **830** | 840 | 0.002022 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **831** | 80 | 0.002022 | [MAKERS MARK MINI 50ml (10 pack)] |
| **832** | 569 | 0.002022 | [CROWN ROYAL VANILLA 750ml (12 pack), CROWN RO... |
| **833** | 754 | 0.002019 | [CHI-CHI'S SKINNY MARGARITA 1750ml (6 pack), C... |
| **834** | 371 | 0.002019 | [[SHANKY''S WHIP 750ml (6 pack), SHANKYS WHIP ... |
| **835** | 726 | 0.002015 | [CAPTAIN MORGAN ORIGINAL SPICED MINI 50ml (12 ... |
| **836** | 765 | 0.002012 | [JOSE CUERVO AUTHENTIC LIGHT LIME MARGARITA 17... |
| **837** | 538 | 0.002012 | [BARTON NATURALS VODKA 1750ml (6 pack), FIREBA... |
| **838** | 108 | 0.002012 | [REVELTON HONEY WHISKEY 750ml (6 pack)] |
| **839** | 621 | 0.002008 | [CAPTAIN MORGAN ORIGINAL SPICED MINI 50ml (12 ... |

| | index | support | itemsets |
|---|---|---|---|
| **840** | 773 | 0.002008 | [FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml... |
| **841** | 167 | 0.002008 | [ABSOLUT MANDRIN 750ml (12 pack)] |
| **842** | 585 | 0.002005 | [CROWN ROYAL REGAL APPLE 750ml (12 pack), CROW... |
| **843** | 549 | 0.002005 | [RYANS CREAM LIQUEUR 1750ml (6 pack), FIREBALL... |
| **844** | 705 | 0.002005 | [HAWKEYE VODKA PET 750ml (12 pack), PARAMOUNT ... |
| **845** | 793 | 0.002005 | [GRANGALA TRIPLE ORANGE LIQUEUR 750ml (12 pack... |

### Association Rules

Association Rules will give an understanding of frequent item sets by assigning different scores based on different criteria. The min threshold parameter used above will be added so that the rarely purchased items are excluded.

In [ ]:
```
apriori_assocs = association_rules(item_sets, min_threshold=min_support)
```

Confidence and Lift are two commonly used metrics for evaluating association rules. Confidence is the probability that the association rule will result in a transaction. Therefore, the higher the confidence, the higher the chance that the rule is a good prediction.

Lift is the ratio at which two item sets appear together than what would be expected if they were statistically independent. The higher the Lift, the higher the probability that the association rule is a good prediction.
In the next section, these metrics will be used to evaluate the best outcomes.

### Distributions of Lift and Confidence

In [ ]:
```
from sklearn.preprocessing import StandardScaler
import numpy as np
from sklearn.preprocessing import power_transform

scaler = StandardScaler()
```

In [ ]:
```
def get_assoc_score_distrs(data, columns, scaling_method="standard_scaler"):
    scaled_cols = [f"{col}_scaled" for col in columns]

    if scaling_method == "standard_scaler":
      data[scaled_cols] = scaler.fit_transform(data[columns])

    elif scaling_method == "box_cox":
      data[scaled_cols] = power_transform(data[columns], method='box-cox')

    else:
      data[scaled_cols] = (data[columns] - data[columns].mean(axis=0))/ data[columns].std(axis=0)
    return data

conf_lift_distr = get_assoc_score_distrs(apriori_assocs[["confidence", "lift"]],
                                         ["confidence", "lift"],
                                         scaling_method="custom")
conf_lift_distr.head(5)
```

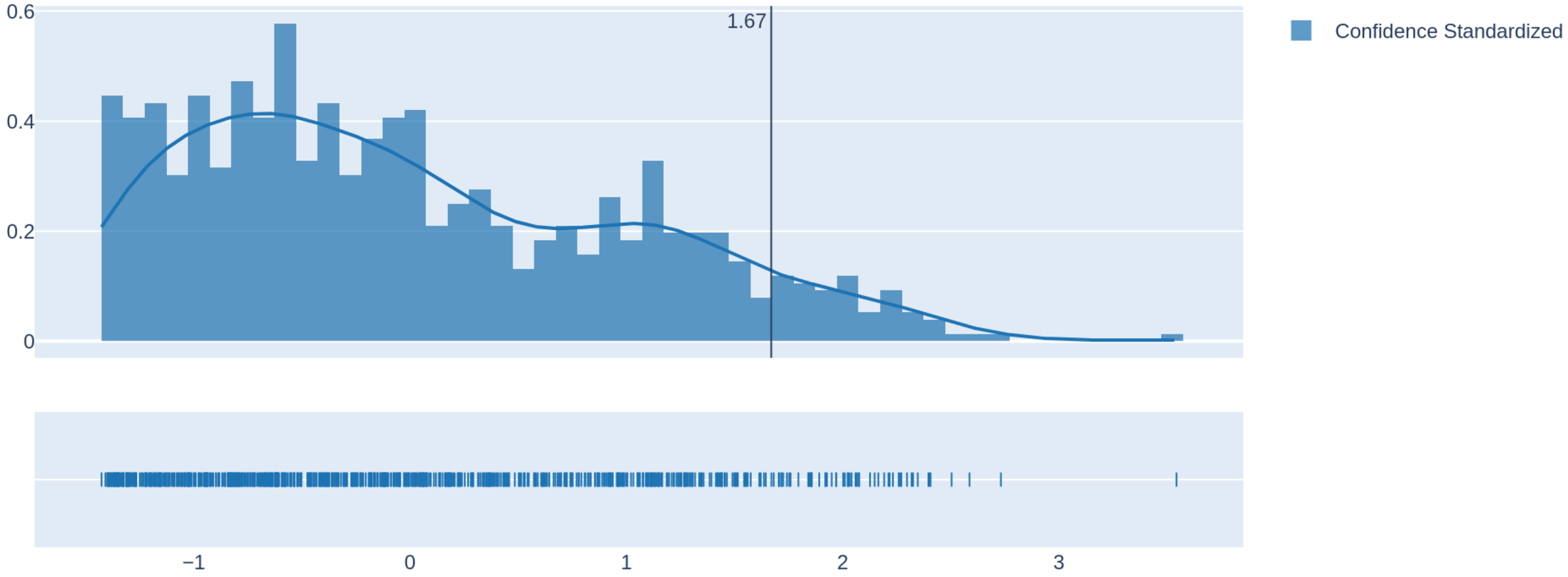
Out[ ]:

| | confidence | lift | confidence_scaled | lift_scaled |
|---|---|---|---|---|
| **0** | 0.484919 | 24.873879 | 0.990930 | -0.526774 |
| **1** | 0.110602 | 24.873879 | -1.352368 | -0.526774 |
| **2** | 0.574633 | 28.874719 | 1.552556 | -0.244466 |
| **3** | 0.128391 | 28.874719 | -1.241001 | -0.244466 |
| **4** | 0.426136 | 36.201174 | 0.622941 | 0.272506 |

In [ ]:

```
hist_data = [conf_lift_distr["confidence_scaled"]]
group_labels = ['Confidence Standardized'] # name of the dataset

v_lines=[1.67]
pyplot_distplot(hist_data, group_labels, v_lines, bin_size=0.1)
```

Out[ ]:




From the distribution chart above, the scaled value (z-score) of 1.67 appears to be a good cutoff point. In this case the outliers at the upper extremities of the distribution are desirable.

We will use this z-score to calculate the minimum value needed for the original data using some simple algebra.

In [ ]:
```
## based on z = x-mu/sigma
apriori_confidence_thresh =  1.67 * (apriori_assocs.confidence.std()) + apriori_assocs.confidence.std()
apriori_confidence_thresh
```

Out[ ]:
```
0.4265044568499144
```

In [ ]:
```
filter = (apriori_assocs.confidence> apriori_confidence_thresh)
top_by_lift_conf = apriori_assocs[filter].sort_values(by=["confidence", "lift"], ascending=False)
len(top_by_lift_conf)
```

Out[ ]:
```
210
```

The next step will find a threshold for lift.

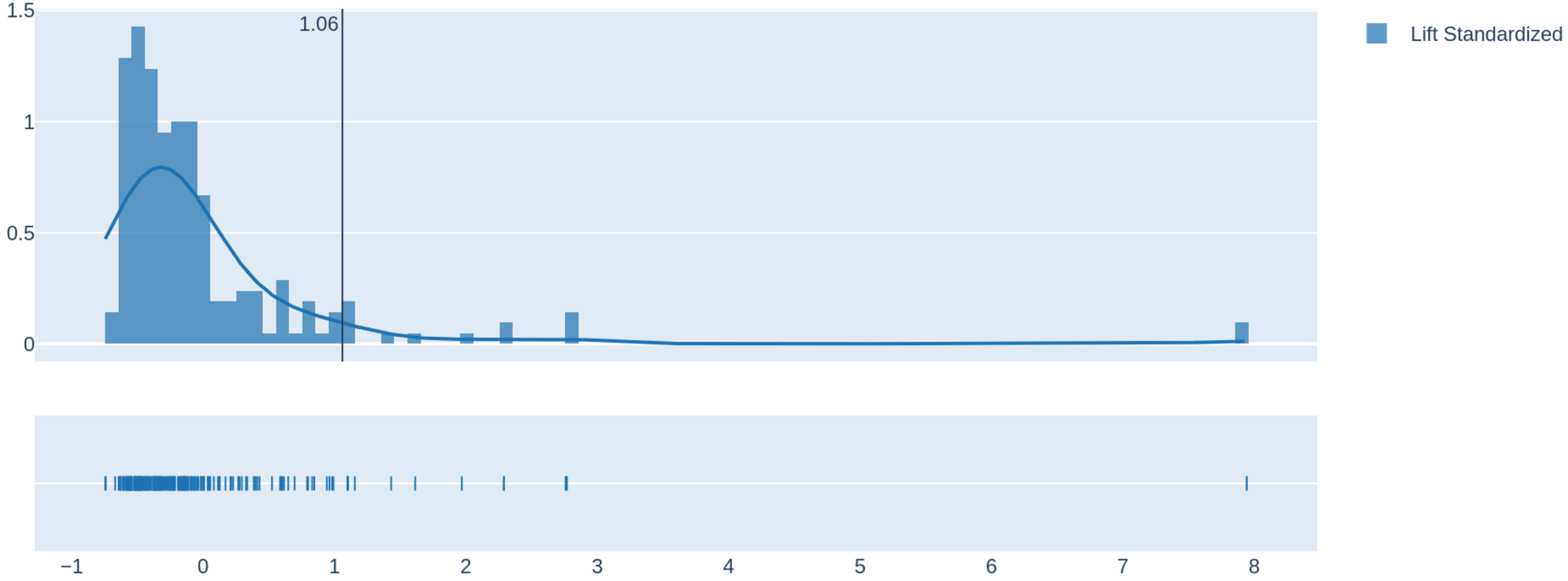
In [ ]:
```
lift_distr = get_assoc_score_distrs(top_by_lift_conf[["lift"]],
                                    ["lift"],
                                    scaling_method="custom")
lift_distr.head(5)
```

Out[ ]:

| | lift | lift_scaled |
|---|---|---|
| **251** | 214.523745 | 7.942461 |
| **748** | 39.136882 | 0.123828 |
| **750** | 37.173766 | 0.036314 |
| **587** | 36.509044 | 0.006681 |
| **568** | 36.471012 | 0.004985 |

In [ ]:
```
hist_data = [lift_distr["lift_scaled"]]
group_labels = ['Lift Standardized'] # name of the dataset
v_lines=[1.06]
pyplot_distplot(hist_data, group_labels, v_lines, bin_size=0.1)
```

Out[ ]:




The Distribution plot suggests that 1.06 is a good cutoff point.

In [ ]:
```
## based on z = x-mu/sigma
apriori_lift_thresh =  1.06 * (apriori_assocs.lift.std()) + apriori_assocs.lift.std()
apriori_lift_thresh
```

Out[ ]:
```
29.194040215264227
```

In [ ]:
```
filter = top_by_lift_conf.lift >= apriori_lift_thresh
top_by_lift_conf = top_by_lift_conf[filter].sort_values(by=["confidence", "lift"], ascending=False)

len(top_by_lift_conf)
```

Out[ ]: 117

This reduced the association rules where reduced significantly.

In [ ]:
```
a_item_name_ids = [list(v) for v in top_by_lift_conf.antecedents.values]
c_item_name_ids = [list(v) for v in top_by_lift_conf.consequents.values]

top_by_lift_conf["antecedents_name"] = [list(product_names.product_name.loc[l]) for l in a_item_name_ids]
top_by_lift_conf["consequent_name"]  = [list(product_names.product_name.loc[l]) for l in c_item_name_ids]
```

This section will print the first 10 association rules based on Confidence and Lift.

In [ ]:
```
for i,v in top_by_lift_conf[["antecedents_name","consequent_name", "confidence", "lift"]].head(10).iterrows():
  print("Antecedent:  " , v[0])
  print("Consequent:  " , v[1])
  print("Confidence:  ",  v[2])
  print("Lift:        ", v[3] , "\n ---------------------------------")
```

```
Antecedent:   ['BLANTONS BOURBON 750ml (6 pack)']
Consequent:   ['BUFFALO TRACE BOURBON 750ml (12 pack)']
Confidence:   0.8926470588235295
Lift:         214.5237445308702
 ---------------------------------
Antecedent:   ['FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml (1 pack)', 'FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (12 pack)', 'FIREBALL CINNAMON WHISKEY PARTY BUCKET 50ml (1 pack)']
Consequent:   ['FIREBALL CINNAMON WHISKEY 100ml (48 pack)']
Confidence:   0.762977473065622
Lift:         39.13688170227999
 ---------------------------------
Antecedent:   ['FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml (1 pack)', 'FIREBALL CINNAMON WHISKEY PARTY BUCKET 50ml (1 pack)', 'FIREBALL CINNAMON WHISKEY 100ml (48 pack)']
Consequent:   ['FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (12 pack)']
Confidence:   0.7397910731244065
Lift:         37.173766363386775
 ---------------------------------
Antecedent:   ['FIREBALL CINNAMON WHISKEY 100ML CARRIER 100ml (8 pack)', 'FIREBALL CINNAMON WHISKEY 100ml (48 pack)']
Consequent:   ['FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (12 pack)']
Confidence:   0.7265625
Lift:         36.50904370787973
 ---------------------------------
Antecedent:   ['FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml (1 pack)', 'FIREBALL CINNAMON WHISKEY PARTY BUCKET 50ml (1 pack)']
Consequent:   ['FIREBALL CINNAMON WHISKEY 100ml (48 pack)']
Confidence:   0.7110060769750168
Lift:         36.471012194327415
 ---------------------------------
Antecedent:   ['FIREBALL CINNAMON WHISKEY 200ml (48 pack)', 'FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (12 pack)']
Consequent:   ['FIREBALL CINNAMON WHISKEY 100ml (48 pack)']
Confidence:   0.7102908277404922
Lift:         36.434323529632294
 ---------------------------------
Antecedent:   ['CAPTAIN MORGAN ORIGINAL SPICED 1000ml (12 pack)', 'CROWN ROYAL REGAL APPLE 1000ml (12 pack)']
Consequent:   ['CROWN ROYAL 1000ml (12 pack)']
Confidence:   0.709480122324159
Lift:         98.24306392893548
 ---------------------------------
Antecedent:   ['FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml (1 pack)', 'FIREBALL CINNAMON WHISKEY 375ml (24 pack)']
Consequent:   ['FIREBALL CINNAMON WHISKEY 100ml (48 pack)']
Confidence:   0.7015968063872255
Lift:         35.98836424874717
 ---------------------------------
Antecedent:   ['CROWN ROYAL 1000ml (12 pack)', 'CROWN ROYAL REGAL APPLE 1000ml (12 pack)']
Consequent:   ['CAPTAIN MORGAN ORIGINAL SPICED 1000ml (12 pack)']
Confidence:   0.6980942828485457
Lift:         68.46527305356204
 ---------------------------------
Antecedent:   ['BLACK VELVET RESERVE 1750ml (6 pack)']
Consequent:   ['BLACK VELVET 1750ml (6 pack)']
Confidence:   0.6970138383102694
Lift:         30.793884997885378
 ---------------------------------
```

**Market Basket Analysis Conclusion**

Output above shows that when an antecedent is purchased there is a high probability that a consequent will be purchased with it. for example, the output:

***Antecedent:*** ['FIREBALL CINNAMON WHISKEY MINI DISPENSER 50ml (1 pack )', 'FIREBALL CINNAMON WHISKEY 100ml (48 pack )', 'FIREBALL CINNAMON WHISKEY PARTY BUCKET 50ml (1 pack )']

***Consequent:*** ['FIREBALL CINNAMON WHISKEY MINI SLEEVE 50ml (12 pack )']

, indicates that when the three products in the Antecedent are purchased together, the product in the consequent is likely to be purchased with them.

For wholesale buying, this could be a way to find suppliers or distributors with the best offers for this combination of products, if the products can be supplied by different vendors.

# References


[1] Pandas - https://pandas.pydata.org/docs/index.html

[2] Google Big Frames: https://cloud.google.com/python/docs/reference/bigframes/latest/bigframes.dataframe.DataFrame#bigframes_dataframe_DataFrame_to_pandas

[3] Null Mask: https://saturncloud.io/blog/python-pandas-selecting-rows-whose-column-value-is-null-none-nan/#:~:text=The%20simplest%20way%20to%20select,rows%20that%20have%20null%20values.

[4] Numpy - https://numpy.org

[5] Plotly -https://plotly.com/

[6] Random Shuffle Numpy - https://stackoverflow.com/questions/49000093/np-random-shuffle-not-workingpython

[7] Pandas Time difference - https://www.statology.org/pandas-time-difference/

[8] Exponential Smoothing - https://www.statsmodels.org/dev/examples/notebooks/generated/exponential_smoothing.html

[9] StatsModel Convergence Warning - https://stackoverflow.com/questions/61925579/convergencewarning-when-trying-to-use-exponentialsmoothing-to-forecast

[10] Holt-Winters - https://otexts.com/fpp2/holt-winters.html

[11] Exponential Smoothing With Trend and Seasonality - https://www.stat.berkeley.edu/~ryantibs/timeseries-f23/lectures/ets.pdf

[12] Pandas Epoch to DataTime - https://www.statology.org/pandas-convert-epoch-to-datetime/