

```
In [4]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import math
from sklearn.preprocessing import MinMaxScaler
import plotly.express as px
import plotly.graph_objects as go
import numpy as np

# URL of the CSV file
url = "https://storage.googleapis.com/courses_data/Assignment%20CSV/finance_liquor_sales.csv"

# Read the CSV file into a DataFrame
df = pd.read_csv(url)

# Display the first few rows of the DataFrame
print(df)
```

	invoice_and_item_number	date	store_number	\	
0	INV-31797900035	2020-11-10	4967		
1	INV-23548800092	2019-11-27	2601		
2	INV-23609300026	2019-12-02	4962		
3	INV-39482900037	2021-08-24	3719		
4	INV-39520400088	2021-08-25	5423		
..	...	...	...		
195	INV-28352300005	2020-06-29	5292		
196	INV-33910800023	2021-01-29	2644		
197	S30234800064	2016-01-15	4829		
198	S13771300075	2013-08-07	3354		
199	S21330500108	2014-09-22	2560		
	store_name		address	\	
0	Jeff's Market / Blue Grass	102, W Mayne St			
1	Hy-Vee Food Store / Fairfield	1300 West Burlington Ave			
2		Hilltop Grocery	1312 Harrison St		
3	Wal-Mart 0581 / Marshalltown	2802 S Center St			
4		Stammer Liquor Corp	615 2nd Ave		
..	...	...	...		
195		John's Qwik Stop	814, C Ave		
196	Hy-Vee Fort Dodge Wine and Spirits	1511 2nd Ave North			
197		Central City 2	1501 MICHIGAN AVE		
198	Sam's Club 8238 / Davenport	3845 ELMORE AVE.			
199	Hy-Vee Food Store / Marion	3600 BUSINESS HWY 151 EAST			
	city	zip_code	store_location	county_number	\
0	Blue Grass	52726.0	POINT (-90.766126 41.509119)	82.0	
1	Fairfield	52556.0	POINT (-91.978702 41.006456)	51.0	
2	Davenport	52803.0	Nan	82.0	
3	Marshalltown	50158.0	POINT (-92.91231 42.012918)	64.0	
4	Sheldon	51201.0	POINT (-95.854899 43.184242)	71.0	
..	...	...	...	...	
195	Vinton	52349.0	POINT (-92.027865 42.163119)	6.0	
196	Fort Dodge	50501.0	POINT (-94.177165 42.508344)	94.0	
197	DES MOINES	50314.0	POINT (-93.613739 41.60572)	77.0	
198	DAVENPORT	52807.0	POINT (-90.527081 41.559731)	82.0	
199	MARION	52302.0	Nan	57.0	
	county	...	item_number	\	
0	SCOTT	...	26858		
1	JEFFERSON	...	86251		
2	SCOTT	...	41844		
3	MARSHALL	...	65259		
4	OBRIEN	...	77487		
..	...	...	...		
195	BENTON	...	43038		
196	WEBSTER	...	101217		
197	Polk	...	86251		
198	Scott	...	36887		
199	Linn	...	33717		
	item_description	pack	bottle_volume_ml	\	
0	Templeton 4YR Rye	12	375		
1	Juarez Triple Sec	12	1000		
2	Everclear Alcohol	24	375		
3	Jagermeister Liqueur Mini Meisters	12	20		
4	Tortilla Gold DSS	12	1000		
..	...	...	...		
195	Bacardi Gold	6	1750		
196	Captain Morgan OSR PET 6/1.75l w/ 50ml CM Sli...	6	1750		
197	Juarez Triple Sec	12	1000		
198	McCormick Vodka	12	1000		
199	Paramount Sloe Gin	12	1000		

```

state_bottle_cost state_bottle_retail bottles_sold sale_dollars \
0                9.99          14.99        1      14.99
1                2.42          3.63        6     21.78
2                4.20          6.30        3     18.90
3                4.93          7.40       12    88.80
4                4.85          7.28        4     29.12
..
..               ...
195              15.00         22.50        3     67.50
196              18.00         27.00      156   4212.00
197              2.42          3.63       60    217.80
198              4.13          6.20      204  1264.80
199              5.17          7.76        4     31.04

volume_sold_liters volume_sold_gallons
0                  0.37          0.09
1                  6.00          1.58
2                  1.12          0.29
3                  0.24          0.06
4                  4.00          1.05
..
..               ...
195              5.25          1.38
196              273.00         72.11
197              60.00          15.85
198              204.00         53.89
199              4.00          1.06

```

[200 rows x 24 columns]

```
In [5]: # Check the data types of each column in the DataFrame
data_types = df.dtypes

# Display the data types
print(data_types)
```

invoice_and_item_number	object
date	object
store_number	int64
store_name	object
address	object
city	object
zip_code	float64
store_location	object
county_number	float64
county	object
category	float64
category_name	object
vendor_number	float64
vendor_name	object
item_number	int64
item_description	object
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

```
In [6]: # Check for missing values in the DataFrame
missing_values = df.isnull().sum()
```

```
# Display the number of missing values for each column
print(missing_values)
```

```
invoice_and_item_number      0
date                          0
store_number                  0
store_name                    0
address                       0
city                          0
zip_code                      0
store_location                18
county_number                 1
county                        1
category                      3
category_name                 9
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
dtype: int64
```

In [7]: # Calculate the percentage of missing values for each column  
missing\_percentage = (df.isnull().mean() \* 100).round(2)

```
# Display the percentage of missing values for each column
print(missing_percentage)
```

```
invoice_and_item_number      0.0
date                          0.0
store_number                  0.0
store_name                    0.0
address                       0.0
city                          0.0
zip_code                      0.0
store_location                9.0
county_number                 0.5
county                        0.5
category                      1.5
category_name                 4.5
vendor_number                 0.0
vendor_name                   0.0
item_number                   0.0
item_description              0.0
pack                          0.0
bottle_volume_ml             0.0
state_bottle_cost             0.0
state_bottle_retail           0.0
bottles_sold                  0.0
sale_dollars                  0.0
volume_sold_liters            0.0
volume_sold_gallons            0.0
dtype: float64
```

In [8]: # Set the threshold to the number of non-null values you want to keep  
threshold = len(df.columns) - 1 # Set the threshold to Len(df.columns) - 1 for more than 2 null values  
  
# Drop rows with more than 2 null values

```
df_cleaned = df.dropna(axis=0, thresh=threshold)
print(df_cleaned)
# Drop rows with missing values
df_cleaned2 = df_cleaned.dropna(axis=1)

# Display the first few rows of the cleaned DataFrame
print(df_cleaned2)
df=df_cleaned2.copy()
```

	invoice_and_item_number	date	store_number	\	
0	INV-31797900035	2020-11-10	4967		
1	INV-23548800092	2019-11-27	2601		
2	INV-23609300026	2019-12-02	4962		
3	INV-39482900037	2021-08-24	3719		
4	INV-39520400088	2021-08-25	5423		
..	...	...	...		
195	INV-28352300005	2020-06-29	5292		
196	INV-33910800023	2021-01-29	2644		
197	S30234800064	2016-01-15	4829		
198	S13771300075	2013-08-07	3354		
199	S21330500108	2014-09-22	2560		
	store_name		address	\	
0	Jeff's Market / Blue Grass	102, W Mayne St			
1	Hy-Vee Food Store / Fairfield	1300 West Burlington Ave			
2		Hilltop Grocery	1312 Harrison St		
3	Wal-Mart 0581 / Marshalltown	2802 S Center St			
4		Stammer Liquor Corp	615 2nd Ave		
..	...	...	...		
195		John's Qwik Stop	814, C Ave		
196	Hy-Vee Fort Dodge Wine and Spirits	1511 2nd Ave North			
197		Central City 2	1501 MICHIGAN AVE		
198	Sam's Club 8238 / Davenport	3845 ELMORE AVE.			
199	Hy-Vee Food Store / Marion	3600 BUSINESS HWY 151 EAST			
	city	zip_code	store_location	county_number	\
0	Blue Grass	52726.0	POINT (-90.766126 41.509119)	82.0	
1	Fairfield	52556.0	POINT (-91.978702 41.006456)	51.0	
2	Davenport	52803.0	Nan	82.0	
3	Marshalltown	50158.0	POINT (-92.91231 42.012918)	64.0	
4	Sheldon	51201.0	POINT (-95.854899 43.184242)	71.0	
..	...	...	...	...	
195	Vinton	52349.0	POINT (-92.027865 42.163119)	6.0	
196	Fort Dodge	50501.0	POINT (-94.177165 42.508344)	94.0	
197	DES MOINES	50314.0	POINT (-93.613739 41.60572)	77.0	
198	DAVENPORT	52807.0	POINT (-90.527081 41.559731)	82.0	
199	MARION	52302.0	Nan	57.0	
	county	...	item_number	\	
0	SCOTT	...	26858		
1	JEFFERSON	...	86251		
2	SCOTT	...	41844		
3	MARSHALL	...	65259		
4	OBRIEN	...	77487		
..	...	...	...		
195	BENTON	...	43038		
196	WEBSTER	...	101217		
197	Polk	...	86251		
198	Scott	...	36887		
199	Linn	...	33717		
	item_description	pack	bottle_volume_ml	\	
0	Templeton 4YR Rye	12	375		
1	Juarez Triple Sec	12	1000		
2	Everclear Alcohol	24	375		
3	Jagermeister Liqueur Mini Meisters	12	20		
4	Tortilla Gold DSS	12	1000		
..	...	...	...		
195	Bacardi Gold	6	1750		
196	Captain Morgan OSR PET 6/1.75l w/ 50ml CM Sli...	6	1750		
197	Juarez Triple Sec	12	1000		
198	McCormick Vodka	12	1000		
199	Paramount Sloe Gin	12	1000		

```
state_bottle_cost state_bottle_retail bottles_sold sale_dollars \
0           9.99        14.99       1     14.99
1           2.42        3.63       6    21.78
2           4.20        6.30       3    18.90
3           4.93        7.40      12   88.80
4           4.85        7.28       4    29.12
..          ...
195         15.00       22.50       3    67.50
196         18.00       27.00      156  4212.00
197         2.42        3.63      60   217.80
198         4.13        6.20      204  1264.80
199         5.17        7.76       4    31.04
```

```
volume_sold_liters volume_sold_gallons
0             0.37        0.09
1             6.00        1.58
2             1.12        0.29
3             0.24        0.06
4             4.00        1.05
..            ...
195          5.25        1.38
196          273.00      72.11
197          60.00        15.85
198          204.00      53.89
199          4.00        1.06
```

[195 rows x 24 columns]

```
invoice_and_item_number date store_number \
0      INV-31797900035 2020-11-10      4967
1      INV-23548800092 2019-11-27      2601
2      INV-23609300026 2019-12-02      4962
3      INV-39482900037 2021-08-24      3719
4      INV-39520400088 2021-08-25      5423
..            ...
195     INV-28352300005 2020-06-29      5292
196     INV-33910800023 2021-01-29      2644
197     S30234800064 2016-01-15      4829
198     S13771300075 2013-08-07      3354
199     S21330500108 2014-09-22      2560
```

```
store_name address \
0      Jeff's Market / Blue Grass      102, W Mayne St
1      Hy-Vee Food Store / Fairfield  1300 West Burlington Ave
2      Hilltop Grocery                1312 Harrison St
3      Wal-Mart 0581 / Marshalltown  2802 S Center St
4      Stammer Liquor Corp          615 2nd Ave
..            ...
195     John's Qwik Stop            814, C Ave
196     Hy-Vee Fort Dodge Wine and Spirits 1511 2nd Ave North
197     Central City 2              1501 MICHIGAN AVE
198     Sam's Club 8238 / Davenport    3845 ELMORE AVE.
199     Hy-Vee Food Store / Marion    3600 BUSINESS HWY 151 EAST
```

```
city zip_code county_number county category ... \
0     Blue Grass  52726.0      82.0    SCOTT 1011600.0 ...
1     Fairfield  52556.0      51.0  JEFFERSON 1081500.0 ...
2     Davenport   52803.0      82.0    SCOTT 1091200.0 ...
3     Marshalltown 50158.0      64.0  MARSHALL 1082100.0 ...
4     Sheldon    51201.0      71.0  OBRIEN 1092100.0 ...
..            ...
195    Vinton    52349.0       6.0    BENTON 1062100.0 ...
196    Fort Dodge 50501.0      94.0  WEBSTER 1700000.0 ...
197    DES MOINES 50314.0      77.0    Polk 1081400.0 ...
```

```

198    DAVENPORT  52807.0      82.0      Scott  1031100.0  ...
199    MARION     52302.0      57.0      Linn   1041200.0  ...

    item_number          item_description  pack  \
0        26858            Templeton 4YR Rye    12
1        86251            Juarez Triple Sec    12
2        41844            Everclear Alcohol    24
3        65259  Jagermeister Liqueur Mini Meisters    12
4        77487            Tortilla Gold DSS    12
..       ...
195      43038            Bacardi Gold      6
196  101217 Captain Morgan OSR PET 6/1.75l w/ 50ml CM Sli...    6
197      86251            Juarez Triple Sec    12
198      36887            McCormick Vodka    12
199      33717            Paramount Sloe Gin    12

    bottle_volume_ml  state_bottle_cost  state_bottle_retail  bottles_sold  \
0           375           9.99           14.99             1
1          1000           2.42           3.63             6
2           375           4.20           6.30             3
3            20           4.93           7.40            12
4          1000           4.85           7.28             4
..       ...
195      1750           15.00           22.50             3
196      1750           18.00           27.00           156
197      1000           2.42           3.63             60
198      1000           4.13           6.20            204
199      1000           5.17           7.76             4

    sale_dollars  volume_sold_liters  volume_sold_gallons
0        14.99           0.37           0.09
1       21.78           6.00           1.58
2       18.90           1.12           0.29
3       88.80           0.24           0.06
4       29.12           4.00           1.05
..       ...
195      67.50           5.25           1.38
196    4212.00          273.00          72.11
197     217.80           60.00          15.85
198   1264.80          204.00          53.89
199     31.04           4.00           1.06

```

[195 rows x 22 columns]

```
In [9]: # Check for missing values in the DataFrame
missing_values = df.isnull().sum()

# Display the number of missing values for each column
print(missing_values)
```

```
invoice_and_item_number    0
date                      0
store_number               0
store_name                 0
address                   0
city                      0
zip_code                  0
county_number              0
county                     0
category                  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
dtype: int64
```

```
In [10]: # Convert the 'date' column to datetime
df['date'] = pd.to_datetime(df['date'])

# Filter rows with date between 2016-01-01 and 2019-12-31
filtered_df = df[(df['date'] >= '2016-01-01') & (df['date'] <= '2019-12-31')]

# Display the result
print(filtered_df.reset_index())
```

index	invoice_and_item_number	date	store_number
0	1 INV-23548800092	2019-11-27	2601
1	2 INV-23609300026	2019-12-02	4962
2	6 S30390600011	2016-01-26	2641
3	7 S30348700047	2016-01-25	3162
4	8 S30466200002	2016-02-01	2633
..	...	...	...
68	185 INV-00002300008	2016-08-29	5244
69	187 INV-24116900112	2019-12-23	2670
70	193 INV-08495400086	2017-11-07	4167
71	194 INV-08632500071	2017-11-14	3385
72	197 S30234800064	2016-01-15	4829

	store_name	address
0	Hy-Vee Food Store / Fairfield	1300 West Burlington Ave
1	Hilltop Grocery	1312 Harrison St
2	Hy-Vee Drugstore / Council Bluffs	757 W BROADWAY
3	Nash Finch / Wholesale Food	807 GRANDVIEW
4	Hy-Vee #3 / BDI / Des Moines	3221 SE 14TH ST
..	...	...
68	The Ox & Wren Spirits and Gifts	708 2nd AVE SE
69	Hy-Vee Food Store / Coralville	2004 8th St
70	Iowa Street Market, Inc.	1256 Iowa St
71	Sam's Club 8162 / Cedar Rapids	2605 Blairs Ferry Rd NE
72	Central City 2	1501 MICHIGAN AVE

	city	zip_code	county_number	county	... item_number
0	Fairfield	52556.0	51.0	JEFFERSON	...
1	Davenport	52803.0	82.0	SCOTT	...
2	COUNCIL BLUFFS	51501.0	78.0	Pottawattamie	...
3	MUSCATINE	52761.0	70.0	Muscatine	...
4	DES MOINES	50320.0	77.0	Polk	...
..	...	...	...	...	...
68	Cresco	52136.0	45.0	HOWARD	...
69	Coralville	52241.0	52.0	JOHNSON	...
70	Dubuque	52001.0	31.0	DUBUQUE	...
71	Cedar Rapids	52402.0	57.0	LINN	...
72	DES MOINES	50314.0	77.0	Polk	...

	item_description	pack	bottle_volume_ml	state_bottle_cost
0	Juarez Triple Sec	12	1000	2.42
1	Everclear Alcohol	24	375	4.20
2	99 Peppermint Mini	10	600	5.94
3	Dekuyper Peachtree	12	1000	7.62
4	Di Amore Quattro Orange	12	1000	9.75
..	...	...	...	...
68	Five O'clock Vodka	12	1000	4.17
69	Maestro Agavero Gold	12	1000	7.00
70	Kamora Coffee Liqueur	12	1000	8.39
71	Montezuma Triple Sec	12	1000	2.13
72	Juarez Triple Sec	12	1000	2.42

	state_bottle_retail	bottles_sold	sale_dollars	volume_sold_liters
0	3.63	6	21.78	6.00
1	6.30	3	18.90	1.12
2	8.91	2	17.82	1.20
3	11.43	4	45.72	4.00
4	14.63	120	1755.60	120.00
..	...	...	...	...
68	6.26	2	75.12	2.00
69	10.50	48	504.00	48.00
70	12.59	4	50.36	4.00
71	3.20	216	691.20	216.00
72	3.63	60	217.80	60.00

```
volume_sold_gallons
0                 1.58
1                 0.29
2                 0.32
3                 1.06
4                31.70
..
68                 ...
69                12.68
70                 1.06
71                57.06
72                15.85
```

[73 rows x 23 columns]

```
In [11]: #df=filtered_df
# Group by 'zip_code' and 'item_number', and sum 'bottles_sold'
sum_by_zip_item = filtered_df.groupby(['zip_code', 'item_number'])['bottles_sold'].sum().reset_index()

# Find the index of the row with the maximum sum of 'bottles_sold' for each 'item_number' within each 'zip_code' group
idxmax_sum = sum_by_zip_item.groupby('zip_code')['bottles_sold'].idxmax()

# Select the corresponding rows from the original DataFrame
result_df = sum_by_zip_item.loc[idxmax_sum].reset_index()

# Display the result
print(result_df)
```

index	zip_code	item_number	bottles_sold
0	50010.0	946574	288
1	50022.0	86507	4
2	50111.0	77805	108
3	50131.0	38089	48
4	50158.0	48099	24
5	50263.0	3135	84
6	50265.0	67526	72
7	50266.0	250	90
8	50314.0	86251	240
9	50316.0	48099	48
10	50317.0	56193	24
11	50320.0	973627	120
12	50327.0	43040	102
13	50401.0	986845	48
14	50461.0	35918	30
15	50501.0	38176	108
16	50588.0	84617	4
17	50662.0	86739	8
18	50701.0	43034	7
19	50702.0	77487	768
20	50703.0	168	180
21	50707.0	15626	60
22	50801.0	43037	5
23	51106.0	67527	240
24	51246.0	82187	5
25	51247.0	86112	6
26	51360.0	77487	48
27	51401.0	27189	18
28	51501.0	86251	48
29	51555.0	45247	2
30	52001.0	67557	4
31	52003.0	43031	5
32	52136.0	35917	2
33	52172.0	67524	1
34	52240.0	86251	60
35	52241.0	65750	48
36	52314.0	75087	1560
37	52338.0	27357	90
38	52402.0	86390	216
39	52411.0	41846	36
40	52556.0	86251	6
41	52601.0	35913	48
42	52627.0	67586	36
43	52732.0	45248	24
44	52761.0	82847	4
45	52803.0	41844	3
46	52804.0	48690	2

In [12]: `from sklearn.preprocessing import MinMaxScaler`

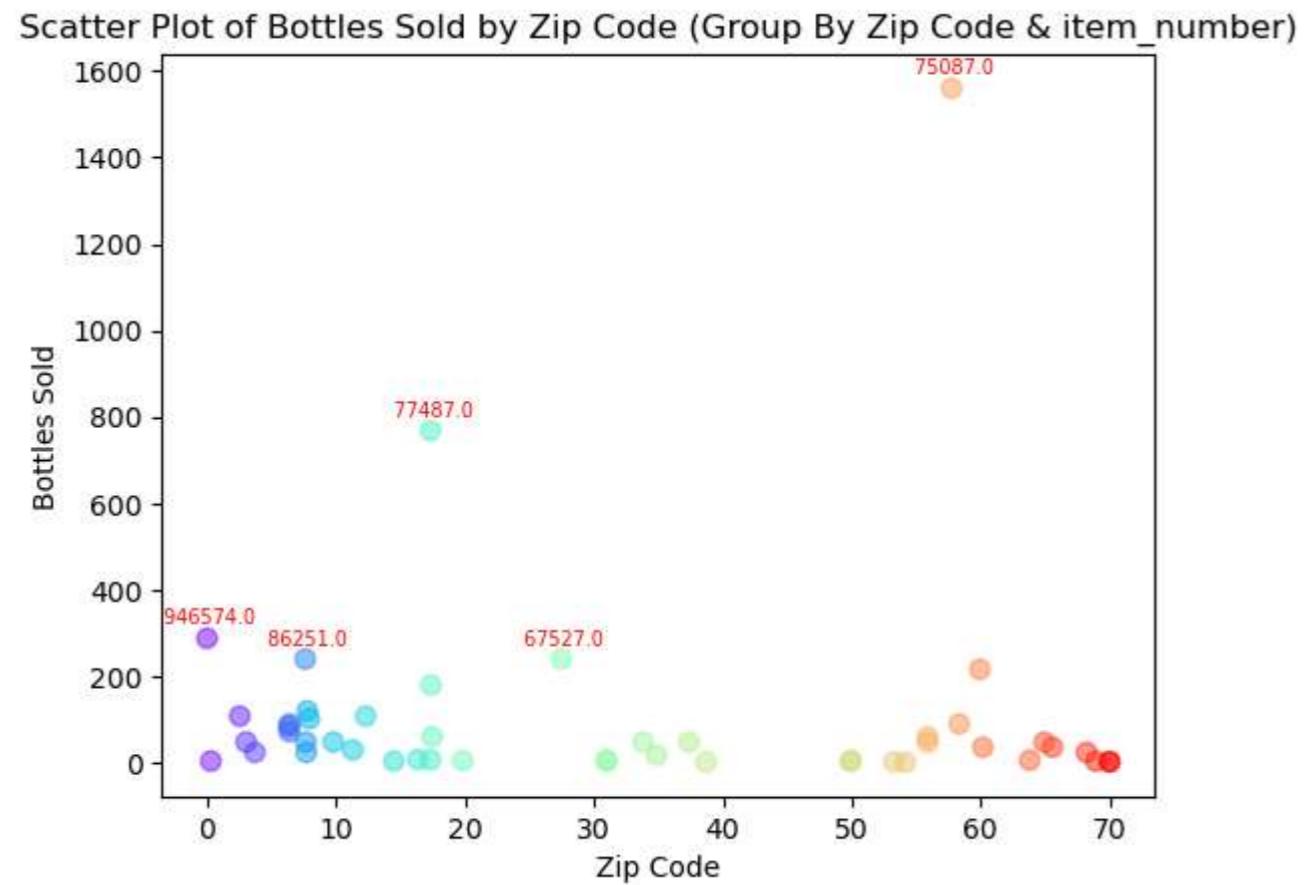
```
# Create a color map with rainbow colors
colors = plt.cm.rainbow(np.linspace(0, 1, len(result_df)))
# Normalize the 'zip_code' values to be between 0 and 75
scaler = MinMaxScaler(feature_range=(0, 70))
result_df['zip_code_normalized'] = scaler.fit_transform(result_df[['zip_code']])
# Identify the top 5 rows based on 'bottles_sold'
top5 = result_df.nlargest(5, 'bottles_sold')

# Scatter plot with rainbow colors
plt.scatter(result_df['zip_code_normalized'], result_df['bottles_sold'], alpha=0.5, c=colors, s=50)
for i, row in top5.iterrows():
    plt.annotate(f" {row['item_number']}")
```

```
(row['zip_code_normalized'], row['bottles_sold']),
textcoords="offset points", xytext=(0,5), ha='center', fontsize=7, color='red')

# Adding labels and title
plt.xlabel('Zip Code')
plt.ylabel('Bottles Sold')
plt.title('Scatter Plot of Bottles Sold by Zip Code (Group By Zip Code & item_number)')

# Display the plot
plt.show()
```



```
In [13]: # Identify the top 5 rows based on 'bottles_sold'
top5 = result_df.nlargest(5, 'bottles_sold')

# Normalize the 'zip_code' values to be between 0 and 75
scaler = MinMaxScaler(feature_range=(0, 70))
result_df['zip_code_normalized'] = scaler.fit_transform(result_df[['zip_code']])

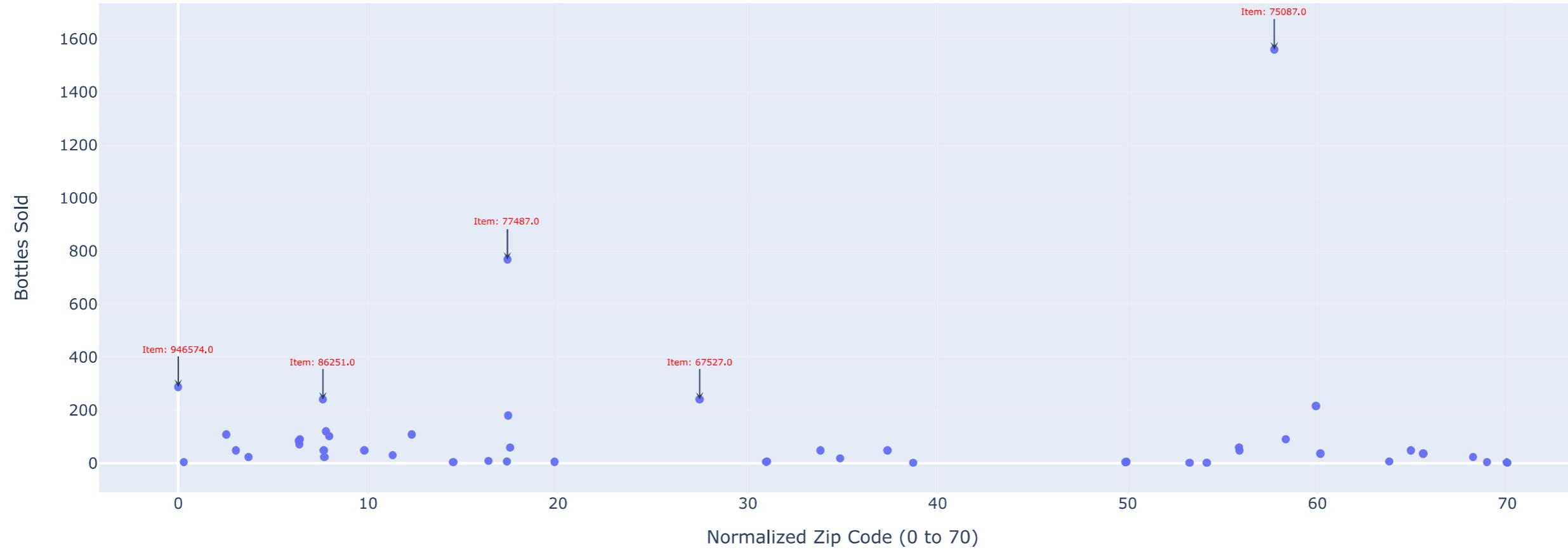
# Scatter plot with rainbow colors using Plotly Express
fig = px.scatter(result_df, x='zip_code_normalized', y='bottles_sold', opacity=0.95)

# Annotate the top 5 points with item numbers only
for i, row in top5.iterrows():
    fig.add_annotation(
        x=row['zip_code_normalized'],
        y=row['bottles_sold'],
        text=f"Item: {row['item_number']}",
        showarrow=True,
        arrowhead=3,
        arrowwidth=1,
        ax=0,
        ay=-30,
        font=dict(size=7, color='red')
    )
```

```
# Updating axis labels and title
fig.update_xaxes(title_text='Normalized Zip Code (0 to 70)')
fig.update_yaxes(title_text='Bottles Sold')
fig.update_layout(title_text='Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code & item_number)',
                 showlegend=False)

# Show the plot
fig.show()
```

Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code &amp; item\_number)



In [14]:

```
# Identify the top 5 rows based on 'bottles_sold'
top5 = result_df.nlargest(5, 'bottles_sold')
```

```
# Normalize the 'zip_code' values to be between 0 and 75
scaler = MinMaxScaler(feature_range=(0, 70))
result_df['zip_code_normalized'] = scaler.fit_transform(result_df[['zip_code']])
```

```
# Thermal scatter plot using Plotly Express
fig = px.scatter(result_df,
                  x='zip_code_normalized',
                  y='bottles_sold',
                  color='bottles_sold',
                  opacity=0.99,
                  color_continuous_scale='thermal',
                  color_discrete_map={'bottles_sold': 'black'},
                  size_max=20)
```

```
# Annotate the top 5 points with item numbers only
for i, row in top5.iterrows():
    fig.add_annotation(
        x=row['zip_code_normalized'],
```

```

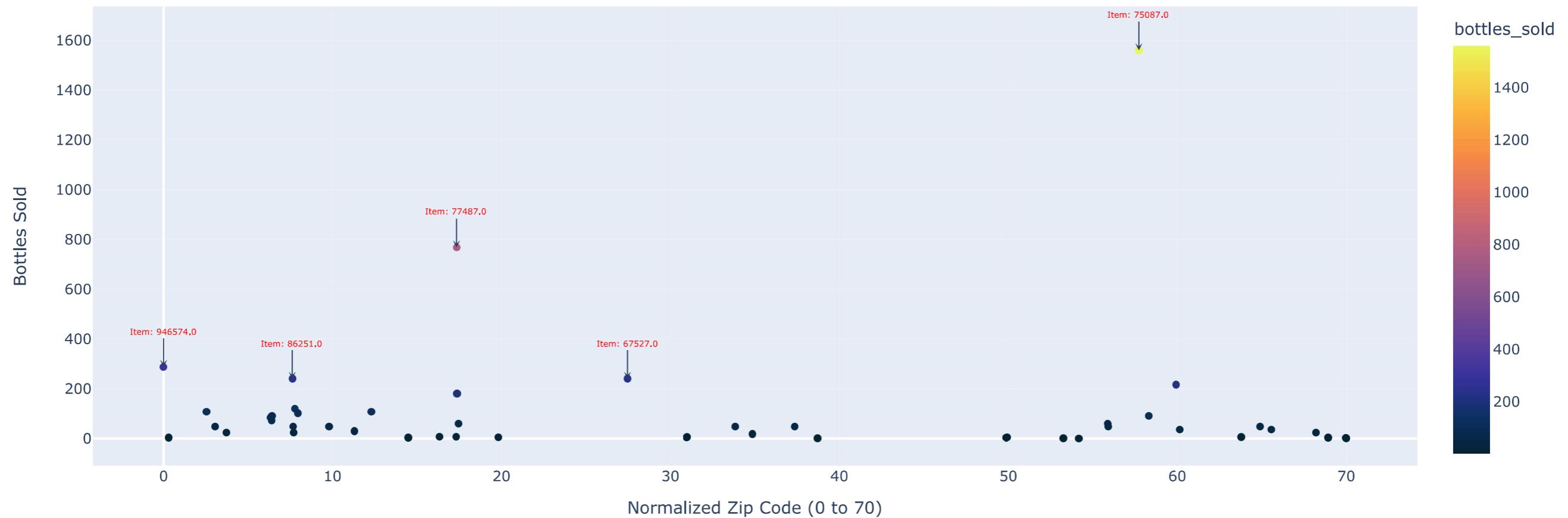
y=row['bottles_sold'],
text=f"Item: {row['item_number']}",
showarrow=True,
arrowhead=3,
arrowwidth=1,
ax=0,
ay=-30,
font=dict(size=7, color='red')
)

# Updating axis labels and title
fig.update_xaxes(title_text='Normalized Zip Code (0 to 70)')
fig.update_yaxes(title_text='Bottles Sold')
fig.update_layout(title_text='Thermal Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code & item_number)',
                  showlegend=False)

# Show the plot
fig.show()

```

Thermal Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code &amp; item\_number)



```

In [15]: # Identify the top 5 rows based on 'bottles_sold'
top5 = result_df.nlargest(5, 'bottles_sold')

# Normalize the 'zip_code' values to be between 0 and 75
scaler = MinMaxScaler(feature_range=(0, 70))
result_df['zip_code_normalized'] = scaler.fit_transform(result_df[['zip_code']])

# Scatter plot with Seaborn
plt.figure(figsize=(10, 6))
sns.scatterplot(x='zip_code_normalized', y='bottles_sold', data=result_df, alpha=0.5,color='blue')

# Annotate the top 5 points with item numbers only

```

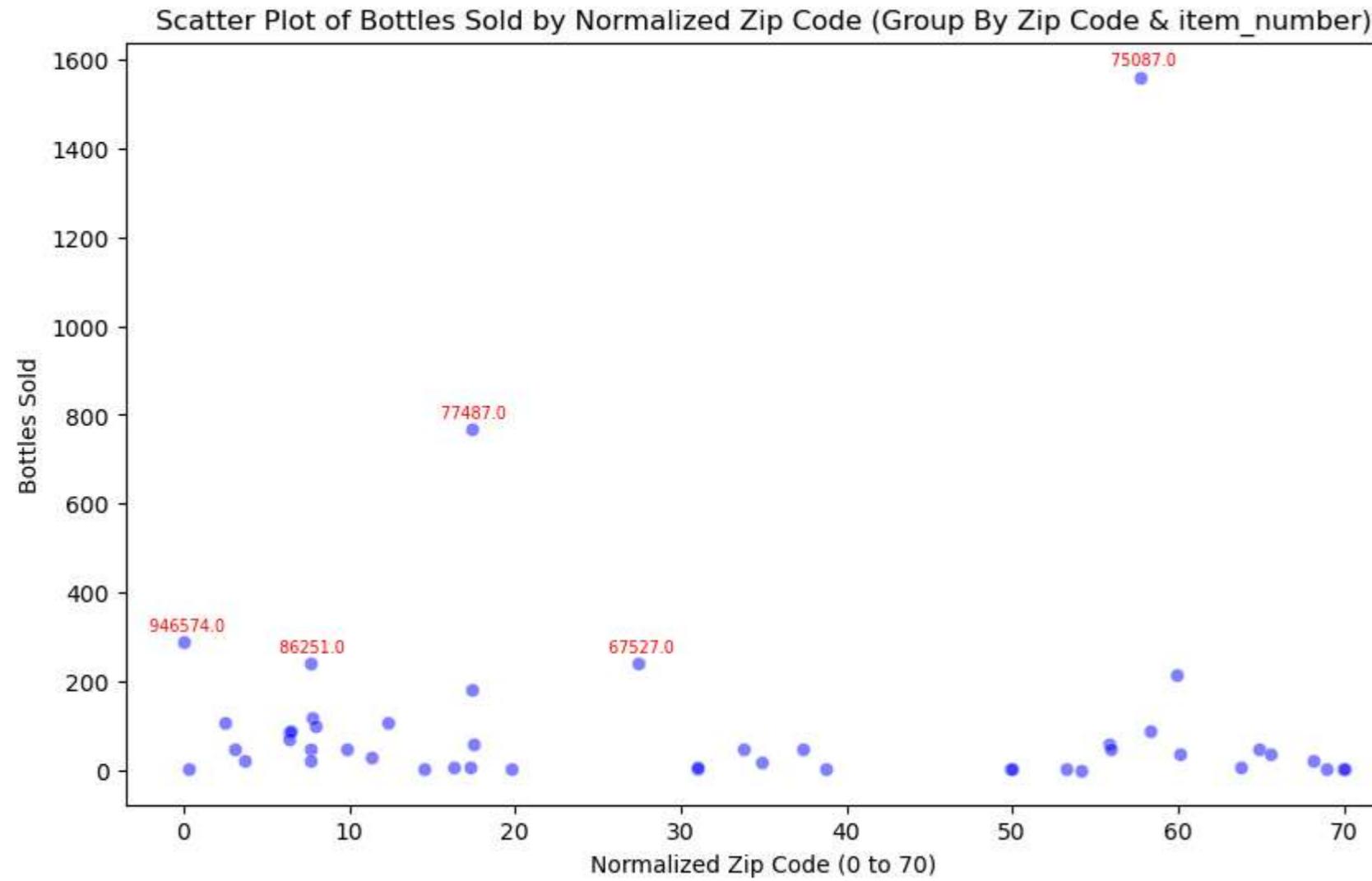
```

for i, row in top5.iterrows():
    plt.annotate(f" {row['item_number']}",
                (row['zip_code_normalized'], row['bottles_sold']),
                textcoords="offset points", xytext=(0, 5), ha='center', fontsize=7, color='red')

# Adding Labels and title
plt.xlabel('Normalized Zip Code (0 to 70)')
plt.ylabel('Bottles Sold')
plt.title('Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code & item_number)')

# Display the plot
plt.show()

```



In [16]:

```

# Assuming result_df is your DataFrame
# 'zip_code' is the column you want to group by

# Group by 'zip_code' and print only the 'zip_code' column
grouped_data1 = filtered_df.groupby('zip_code').size().reset_index(name='count')

# Print only the 'zip_code' column
print(grouped_data1)

```

```
zip_code  count
0    50010.0    1
1    50022.0    1
2    50111.0    1
3    50131.0    1
4    50158.0    3
5    50263.0    1
6    50265.0    1
7    50266.0    1
8    50314.0    4
9    50316.0    1
10   50317.0    1
11   50320.0    3
12   50327.0    1
13   50401.0    2
14   50461.0    1
15   50501.0    1
16   50588.0    1
17   50662.0    1
18   50701.0    3
19   50702.0    2
20   50703.0    4
21   50707.0    1
22   50801.0    1
23   51106.0    3
24   51246.0    1
25   51247.0    1
26   51360.0    1
27   51401.0    3
28   51501.0    4
29   51555.0    1
30   52001.0    1
31   52003.0    1
32   52136.0    1
33   52172.0    1
34   52240.0    2
35   52241.0    2
36   52314.0    2
37   52338.0    1
38   52402.0    2
39   52411.0    1
40   52556.0    1
41   52601.0    2
42   52627.0    1
43   52732.0    1
44   52761.0    1
45   52803.0    1
46   52804.0    1
```

```
In [17]: # Group by 'zip_code' and 'item_number', and sum 'bottles_sold'
grouped_data = filtered_df.groupby(['zip_code','item_number'])['bottles_sold'].sum().reset_index()

# Print the grouped data
print(grouped_data)
```

```

zip_code item_number bottles_sold
0 50010.0 946574 288
1 50022.0 86507 4
2 50111.0 77805 108
3 50131.0 38089 48
4 50158.0 48099 24
.. ...
66 52627.0 67586 36
67 52732.0 45248 24
68 52761.0 82847 4
69 52803.0 41844 3
70 52804.0 48690 2

```

[71 rows x 3 columns]

```

In [18]: # Specify the Excel file path
excel_file_path = 'grouped_data_for_theory1.xlsx'

# Export the grouped data to Excel
grouped_data.to_excel(excel_file_path, index=True)

print(f"Grouped data has been exported to {excel_file_path}")

```

Grouped data has been exported to grouped\_data\_for\_theory1.xlsx

```

In [19]: from sklearn.preprocessing import MinMaxScaler

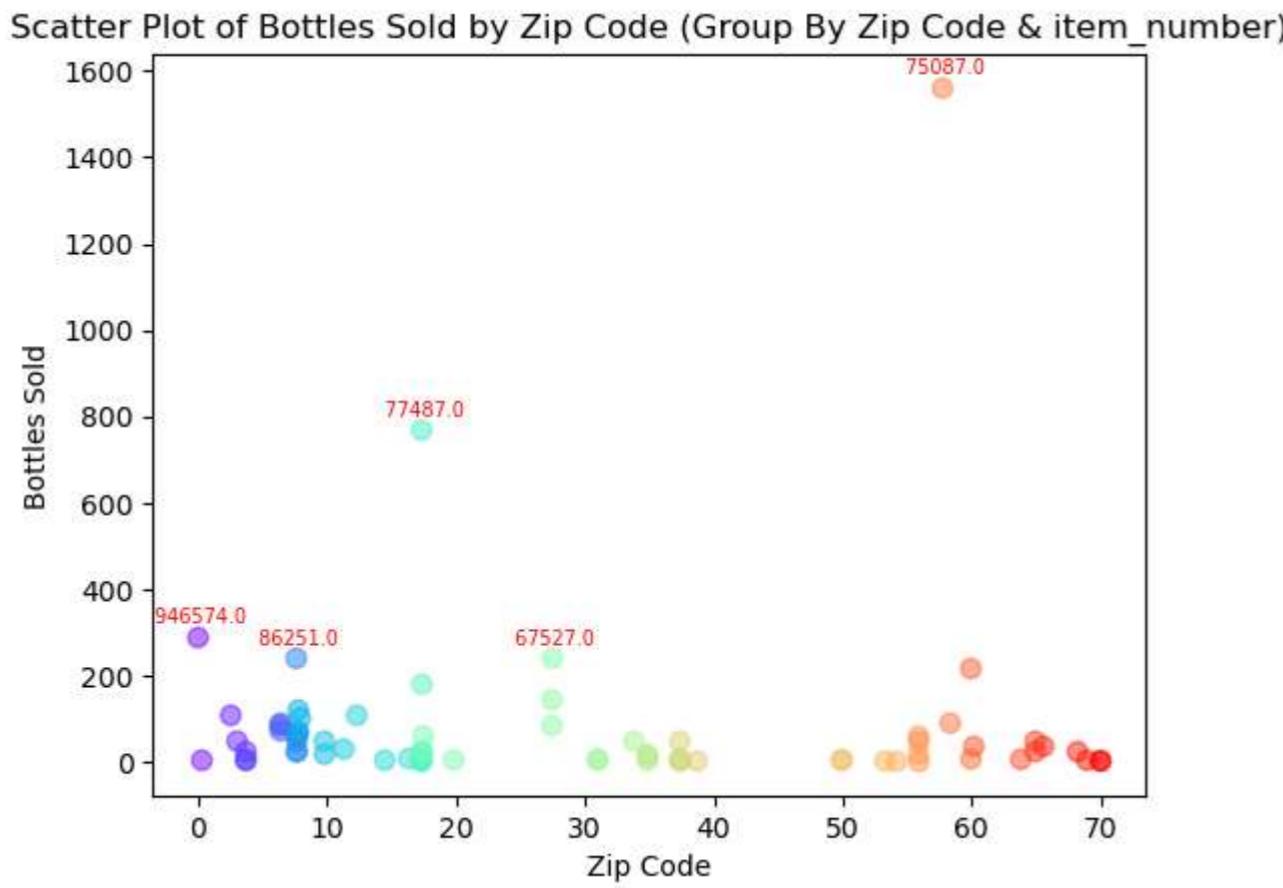
# Create a color map with rainbow colors
colors = plt.cm.rainbow(np.linspace(0, 1, len(grouped_data)))
# Normalize the 'zip_code' values to be between 0 and 75
scaler = MinMaxScaler(feature_range=(0, 75))
grouped_data['zip_code_normalized'] = scaler.fit_transform(grouped_data[['zip_code']])
# Identify the top 5 rows based on 'bottles_sold'
top5 = grouped_data.nlargest(5, 'bottles_sold')

# Scatter plot with rainbow colors
plt.scatter(grouped_data['zip_code_normalized'], grouped_data['bottles_sold'], alpha=0.5, c=colors, s=50)
for i, row in top5.iterrows():
    plt.annotate(f" {row['item_number']}",
                (row['zip_code_normalized'], row['bottles_sold']),
                textcoords="offset points", xytext=(0,5), ha='center', fontsize=7, color='red')

# Adding labels and title
plt.xlabel('Zip Code')
plt.ylabel('Bottles Sold')
plt.title('Scatter Plot of Bottles Sold by Zip Code (Group By Zip Code & item_number)')

# Display the plot
plt.show()

```



```
In [20]: # Identify the top 5 rows based on 'bottles_sold'
top5 = grouped_data.nlargest(5, 'bottles_sold')

# Normalize the 'zip_code' values to be between 0 and 75
scaler = MinMaxScaler(feature_range=(0, 70))
grouped_data['zip_code_normalized'] = scaler.fit_transform(grouped_data[['zip_code']])

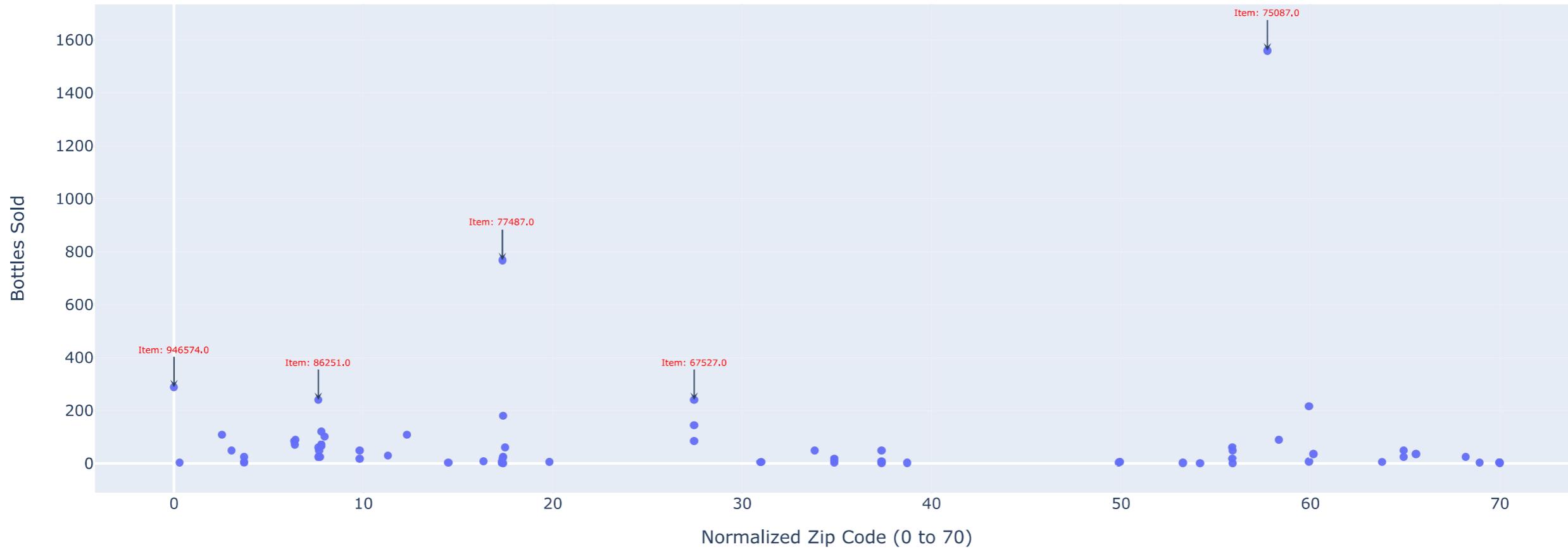
# Scatter plot with rainbow colors using Plotly Express
fig = px.scatter(grouped_data, x='zip_code_normalized', y='bottles_sold', opacity=0.95)

# Annotate the top 5 points with item numbers only
for i, row in top5.iterrows():
    fig.add_annotation(
        x=row['zip_code_normalized'],
        y=row['bottles_sold'],
        text=f"Item: {row['item_number']}",
        showarrow=True,
        arrowhead=3,
        arrowwidth=1,
        ax=0,
        ay=-30,
        font=dict(size=7, color='red')
    )

# Updating axis labels and title
fig.update_xaxes(title_text='Normalized Zip Code (0 to 70)')
fig.update_yaxes(title_text='Bottles Sold')
fig.update_layout(title_text='Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code & item_number)',
                  showlegend=False)

# Show the plot
fig.show()
```

## Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code &amp; item\_number)



```
In [21]: # Identify the top 5 rows based on 'bottles_sold'
top5 = grouped_data.nlargest(5, 'bottles_sold')

# Normalize the 'zip_code' values to be between 0 and 75
scaler = MinMaxScaler(feature_range=(0, 70))
grouped_data['zip_code_normalized'] = scaler.fit_transform(grouped_data[['zip_code']])

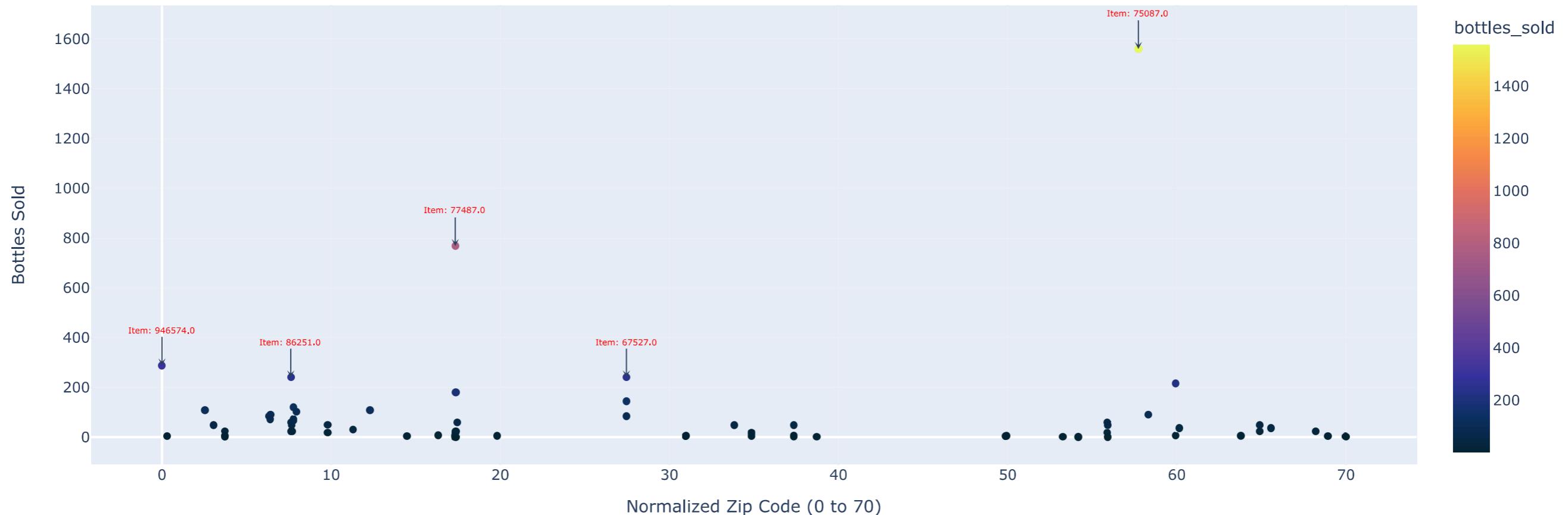
# Thermal scatter plot using Plotly Express
fig = px.scatter(grouped_data,
                  x='zip_code_normalized',
                  y='bottles_sold',
                  color='bottles_sold',
                  opacity=0.99,
                  color_continuous_scale='thermal',
                  color_discrete_map={'bottles_sold': 'black'},
                  size_max=20)

# Annotate the top 5 points with item numbers only
for i, row in top5.iterrows():
    fig.add_annotation(
        x=row['zip_code_normalized'],
        y=row['bottles_sold'],
        text=f"Item: {row['item_number']}",
        showarrow=True,
        arrowhead=3,
        arrowwidth=1,
        ax=0,
        ay=-30,
        font=dict(size=7, color='red')
    )
)
```

```
# Updating axis labels and title
fig.update_xaxes(title_text='Normalized Zip Code (0 to 70)')
fig.update_yaxes(title_text='Bottles Sold')
fig.update_layout(title_text='Thermal Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code & item_number)',
                 showlegend=False)

# Show the plot
fig.show()
```

Thermal Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code &amp; item\_number)



```
In [22]: # Identify the top 5 rows based on 'bottles_sold'
top5 = grouped_data.nlargest(5, 'bottles_sold')

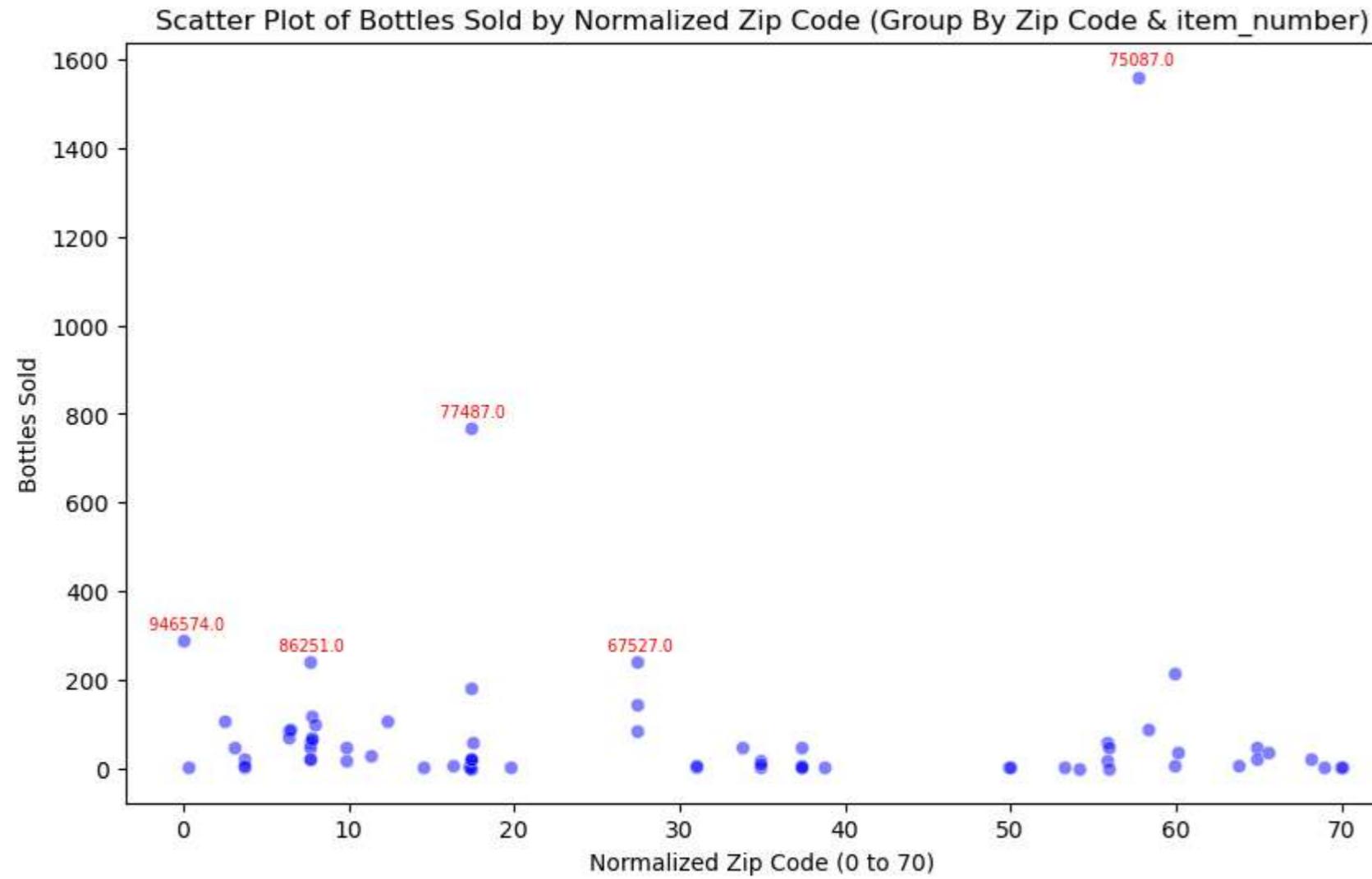
# Normalize the 'zip_code' values to be between 0 and 75
scaler = MinMaxScaler(feature_range=(0, 70))
grouped_data['zip_code_normalized'] = scaler.fit_transform(grouped_data[['zip_code']])

# Scatter plot with Seaborn
plt.figure(figsize=(10, 6))
sns.scatterplot(x='zip_code_normalized', y='bottles_sold', data=grouped_data, alpha=0.5, color='blue')

# Annotate the top 5 points with item numbers only
for i, row in top5.iterrows():
    plt.annotate(f" {row['item_number']}",
                (row['zip_code_normalized'], row['bottles_sold']),
                textcoords="offset points", xytext=(0, 5), ha='center', fontsize=7, color='red')

# Adding Labels and title
plt.xlabel('Normalized Zip Code (0 to 70)')
plt.ylabel('Bottles Sold')
plt.title('Scatter Plot of Bottles Sold by Normalized Zip Code (Group By Zip Code & item_number)')
```

```
# Display the plot  
plt.show()
```



In [23]:

```
# Display the filter data  
print(filtered_df.reset_index())
```

index	invoice_and_item_number	date	store_number
0	1 INV-23548800092	2019-11-27	2601
1	2 INV-23609300026	2019-12-02	4962
2	6 S30390600011	2016-01-26	2641
3	7 S30348700047	2016-01-25	3162
4	8 S30466200002	2016-02-01	2633
..	...	...	...
68	185 INV-00002300008	2016-08-29	5244
69	187 INV-24116900112	2019-12-23	2670
70	193 INV-08495400086	2017-11-07	4167
71	194 INV-08632500071	2017-11-14	3385
72	197 S30234800064	2016-01-15	4829

	store_name	address
0	Hy-Vee Food Store / Fairfield	1300 West Burlington Ave
1	Hilltop Grocery	1312 Harrison St
2	Hy-Vee Drugstore / Council Bluffs	757 W BROADWAY
3	Nash Finch / Wholesale Food	807 GRANDVIEW
4	Hy-Vee #3 / BDI / Des Moines	3221 SE 14TH ST
..	...	...
68	The Ox & Wren Spirits and Gifts	708 2nd AVE SE
69	Hy-Vee Food Store / Coralville	2004 8th St
70	Iowa Street Market, Inc.	1256 Iowa St
71	Sam's Club 8162 / Cedar Rapids	2605 Blairs Ferry Rd NE
72	Central City 2	1501 MICHIGAN AVE

	city	zip_code	county_number	county	... item_number
0	Fairfield	52556.0	51.0	JEFFERSON	...
1	Davenport	52803.0	82.0	SCOTT	...
2	COUNCIL BLUFFS	51501.0	78.0	Pottawattamie	...
3	MUSCATINE	52761.0	70.0	Muscatine	...
4	DES MOINES	50320.0	77.0	Polk	...
..	...	...	...	...	...
68	Cresco	52136.0	45.0	HOWARD	...
69	Coralville	52241.0	52.0	JOHNSON	...
70	Dubuque	52001.0	31.0	DUBUQUE	...
71	Cedar Rapids	52402.0	57.0	LINN	...
72	DES MOINES	50314.0	77.0	Polk	...

	item_description	pack	bottle_volume_ml	state_bottle_cost
0	Juarez Triple Sec	12	1000	2.42
1	Everclear Alcohol	24	375	4.20
2	99 Peppermint Mini	10	600	5.94
3	Dekuyper Peachtree	12	1000	7.62
4	Di Amore Quattro Orange	12	1000	9.75
..	...	...	...	...
68	Five O'clock Vodka	12	1000	4.17
69	Maestro Agavero Gold	12	1000	7.00
70	Kamora Coffee Liqueur	12	1000	8.39
71	Montezuma Triple Sec	12	1000	2.13
72	Juarez Triple Sec	12	1000	2.42

	state_bottle_retail	bottles_sold	sale_dollars	volume_sold_liters
0	3.63	6	21.78	6.00
1	6.30	3	18.90	1.12
2	8.91	2	17.82	1.20
3	11.43	4	45.72	4.00
4	14.63	120	1755.60	120.00
..	...	...	...	...
68	6.26	2	75.12	2.00
69	10.50	48	504.00	48.00
70	12.59	4	50.36	4.00
71	3.20	216	691.20	216.00
72	3.63	60	217.80	60.00

```
volume_sold_gallons
0              1.58
1              0.29
2              0.32
3              1.06
4             31.70
..
68             ...
69            12.68
70             1.06
71            57.06
72            15.85
```

[73 rows x 23 columns]

```
In [24]: # Group by 'store_name' and sum the 'sale_dollars' column
grouped_df = filtered_df.groupby('store_name')['sale_dollars'].sum().reset_index()

# Display the result
print(grouped_df)
```

	store_name	sale_dollars
0	Bootleggin' Barzini's Fin	6.75
1	Burlington Shell	206.64
2	CVS Pharmacy #8526 / Cedar Rapids	42.00
3	Cedar Ridge Vineyards	3712.50
4	Central City 2	2580.60
..	...	...
56	Speedy Gas N Shop	68.18
57	Tequila's Liquor Store	413.28
58	The Ox & Wren Spirits and Gifts	75.12
59	Tobacco Hut #14 / Council Bluffs	33.24
60	Wilkie Liquors	11620.80

[61 rows x 2 columns]

```
In [25]: # Sort in descending order based on the sum of 'sale_dollars'
grouped_df_sorted = grouped_df.sort_values(by='sale_dollars', ascending=False)

# Display the result
print(grouped_df_sorted)
```

	store_name	sale_dollars
60	Wilkie Liquors	11620.80
50	Sam's Club 6432 / Sioux City	6641.04
51	Sam's Club 6514 / Waterloo	5199.36
18	Hy-Vee #3 / BDI / Des Moines	4124.04
52	Sam's Club 6568 / Ames	3913.92
..	...	...
21	Hy-Vee Drugstore / Council Bluffs	17.82
16	Food Land Super Markets	13.26
6	Double D Liquor Store	9.74
38	Hy-Vee Wine and Spirits / Waterloo	7.00
0	Bootleggin' Barzini's Fin	6.75

[61 rows x 2 columns]

```
In [26]: # Keep only the first 15 rows
top_15_rows = grouped_df_sorted.head(15)

# Display the result
print(top_15_rows)
```

	store_name	sale_dollars
60	Wilkie Liquors	11620.80
50	Sam's Club 6432 / Sioux City	6641.04
51	Sam's Club 6514 / Waterloo	5199.36
18	Hy-Vee #3 / BDI / Des Moines	4124.04
52	Sam's Club 6568 / Ames	3913.92
3	Cedar Ridge Vineyards	3712.50
37	Hy-Vee Wine and Spirits / WDM	3372.30
4	Central City 2	2580.60
10	Fareway Stores #138 / Pleasant Hill	2295.00
23	Hy-Vee Food Store #2 / Waterloo	1992.15
28	Hy-Vee Food Store / Fort Dodge	1563.84
20	Hy-Vee / Waukee	1518.72
9	Fareway Stores #067 / Evansdale	1349.40
11	Fareway Stores #153 / W Des Moines	1349.28
15	Fareway Stores #983 / Grimes	1296.00

```
In [27]: # Find the sum of the 'sale_dollars' column as a float
total_sale_dollars = float(grouped_df_sorted['sale_dollars'].sum())

# Display the result
print(total_sale_dollars)
```

61149.54

```
In [32]: # Use .loc to create a new column by dividing 'sale_dollars' by the total sale dollars
top_15_rows.loc[:, 'sale_dollars_normalized'] = ((top_15_rows['sale_dollars'] / total_sale_dollars) * 100).copy()

# Display the result
print(top_15_rows)
```

	store_name	sale_dollars	sale_dollars_normalized
60	Wilkie Liquors	11620.80	19.003904
50	Sam's Club 6432 / Sioux City	6641.04	10.860327
51	Sam's Club 6514 / Waterloo	5199.36	8.502697
18	Hy-Vee #3 / BDI / Des Moines	4124.04	6.744188
52	Sam's Club 6568 / Ames	3913.92	6.400571
3	Cedar Ridge Vineyards	3712.50	6.071182
37	Hy-Vee Wine and Spirits / WDM	3372.30	5.514841
4	Central City 2	2580.60	4.220146
10	Fareway Stores #138 / Pleasant Hill	2295.00	3.753094
23	Hy-Vee Food Store #2 / Waterloo	1992.15	3.257833
28	Hy-Vee Food Store / Fort Dodge	1563.84	2.557403
20	Hy-Vee / Waukee	1518.72	2.483616
9	Fareway Stores #067 / Evansdale	1349.40	2.206721
11	Fareway Stores #153 / W Des Moines	1349.28	2.206525
15	Fareway Stores #983 / Grimes	1296.00	2.119395

In [29]:

```
# Sort the DataFrame by 'sale_dollars_normalized' in descending order
top_15_rows_sorted = top_15_rows.sort_values(by='sale_dollars_normalized', ascending=True)

# Create a horizontal bar plot
fig, ax = plt.subplots()
bars = ax.barh(top_15_rows_sorted['store_name'], top_15_rows_sorted['sale_dollars_normalized'])
plt.xlabel('Sale Dollars Normalized (%)')
plt.ylabel('Store Name')
plt.title('Horizontal Bar Plot of Sale Dollars Normalized')

# Add percentage numbers to the bars
for bar in bars:
    plt.text(bar.get_width(), bar.get_y() + bar.get_height() / 2, f'{bar.get_width():.3f}%',
             va='center', ha='left')

# Display the plot
plt.show()
```



In [30]:

```
# Sort the DataFrame by 'sale_dollars_normalized' in descending order
top_15_rows_sorted = top_15_rows.sort_values(by='sale_dollars_normalized', ascending=False)

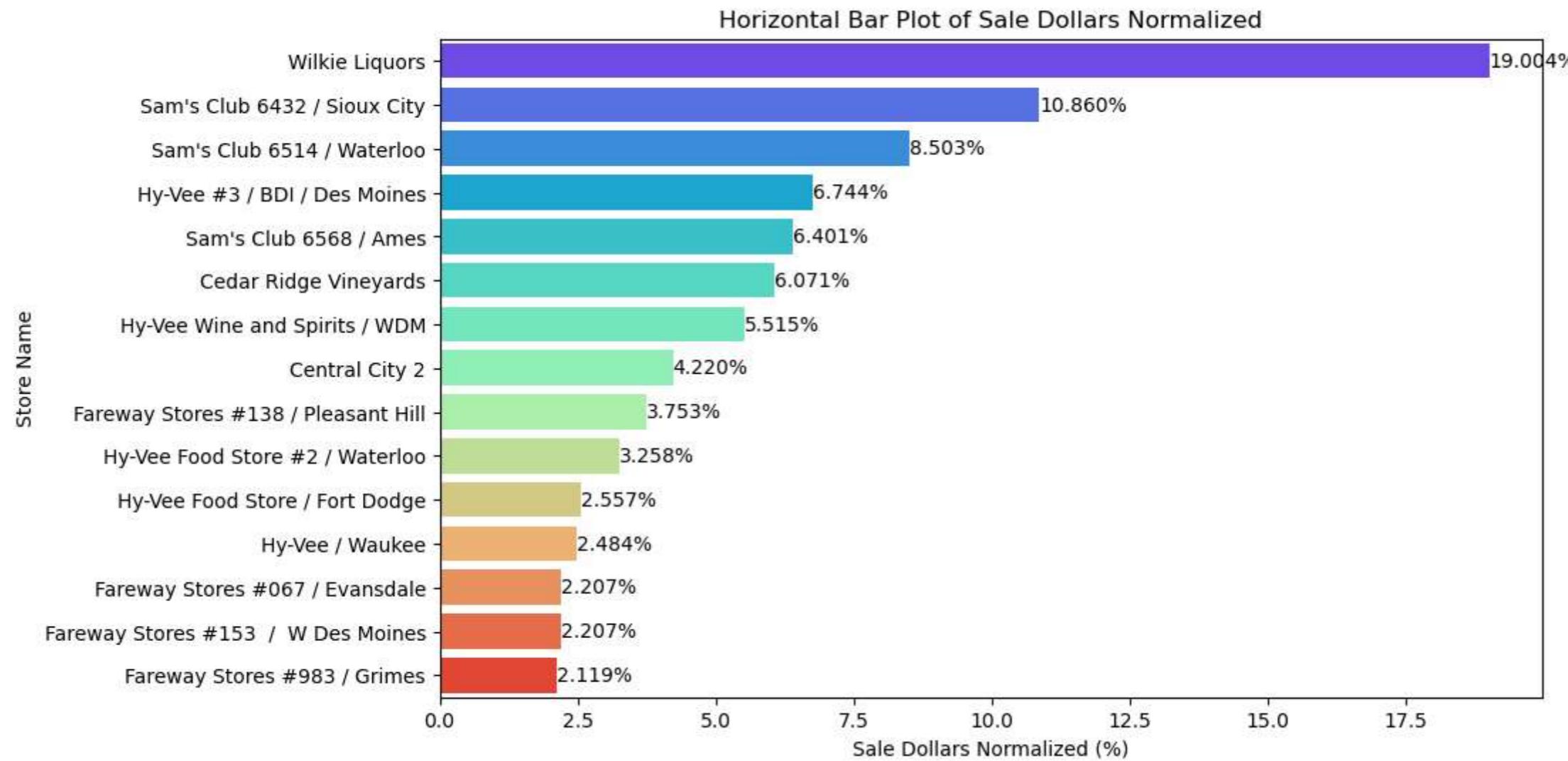
# Create a horizontal bar plot with Seaborn
plt.figure(figsize=(10, 6)) # Adjust the figure size if needed
rainbow_colors = sns.color_palette('rainbow', n_colors=len(top_15_rows_sorted))
ax = sns.barplot(x='sale_dollars_normalized', y='store_name', data=top_15_rows_sorted, palette=rainbow_colors)

# Add percentage numbers to the bars
for p in ax.patches:
    width = p.get_width()
    plt.text(width, p.get_y() + p.get_height() / 2, f'{width:.3f}%', va='center', ha='left')

# Set Labels and title
plt.xlabel('Sale Dollars Normalized (%)')
plt.ylabel('Store Name')
```

```
plt.title('Horizontal Bar Plot of Sale Dollars Normalized')

# Display the plot
plt.show()
```



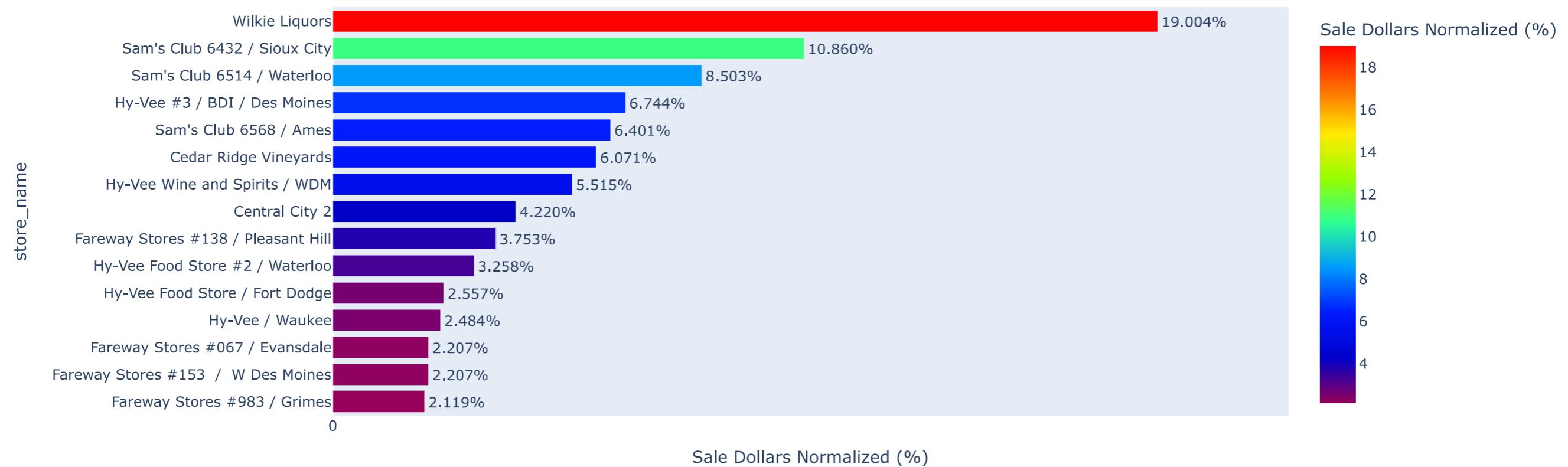
```
In [31]: # Sort the DataFrame by 'sale_dollars_normalized' in descending order
top_15_rows_sorted = top_15_rows.sort_values(by='sale_dollars_normalized', ascending=True)

# Create a horizontal bar plot with Plotly
fig = px.bar(top_15_rows_sorted,
              x='sale_dollars_normalized',
              y='store_name',
              text='sale_dollars_normalized',
              labels={'sale_dollars_normalized': 'Sale Dollars Normalized (%)'},
              title='Horizontal Bar Plot of Sale Dollars Normalized with Plotly',
              color='sale_dollars_normalized',
              color_continuous_scale='Rainbow')

# Adjust the x-axis width
fig.update_layout(xaxis=dict(tickvals=[0, 25, 50, 75, 100], range=[0, 22]))
# Display percentage numbers on the bars
fig.update_traces(texttemplate='%{text:.3f}%', textposition='outside')

# Show the figure
fig.show()
```

## Horizontal Bar Plot of Sale Dollars Normalized with Plotly



In [ ]: