```
#Loading neccesary packages
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
import plotly.express as px


pd.set_option('display.max_columns', None)
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
Mounted at /content/drive
```

```
# Reading the data
# Make Columns (5,17,30,31) as string to avoid the error
#column_types = {5: str, 17: str, 30: str, 31: str}
basket_data_raw = pd.read_csv('/content/drive/MyDrive/Amcon/2021-2023 Years Master(4).csv')
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
<ipython-input-5-c7f3872205e6>:4: DtypeWarning: Columns (11) have mixed types. Specify dtype option on import or set low_mem
  basket_data_raw = pd.read_csv('/content/drive/MyDrive/Amcon/2021-2023 Years Master(4).csv')
```

```
basket_data_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13529472 entries, 0 to 13529471
Data columns (total 29 columns):
 #   Column                       Dtype
---  ------                       -----
 0   Customer.Total.Proper        float64
 1   Transaction.End.Date.Proper  object
 2   Quantity.Sold                float64
 3   Base.Price                   float64
 4   Selling.Price                float64
 5   Department Name Transaction  object
 6   month                        int64
 7   day                          int64
 8   year                         int64
 9   Transaction Store            object
 10  Banner                       object
 11  Item ID                      object
 12  Reciept Alias                object
 13  Item Size                    object
 14  Brand Name                   object
 15  Item Type                    object
 16  Date Created                 object
 17  Category                     object
 18  Subcategory                  object
 19  Anonymous Customer Number    float64
 20  Loyalty Customer?            int64
 21  Opted Into Marketing         int64
 22  Loyalty Balance              float64
 23  Discount receiving?          int64
 24  Customer Number              float64
 25  Receipt Number               float64
 26  Employee Number              float64
 27  Department Number            float64
 28  Sales                        float64
dtypes: float64(11), int64(6), object(12)
memory usage: 2.9+ GB
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

```
basket_data_raw.head()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

	Customer.Total.Proper	Transaction.End.Date.Proper	Quantity.Sold	Base.Price	Selling.Price	Department Name Transaction	month	day	yea
0	3.99	1/14/22	1.00	3.99	3.99	REFRIGERATED	1	14	20
1	6.99	1/14/22	0.27	6.99	1.89	BULK	1	14	20
2	6.99	1/14/22	0.27	6.99	1.89	BULK	1	14	20
3	6.99	1/14/22	0.27	6.99	1.89	BULK	1	14	20
4	9.99	1/11/22	1.19	9.99	11.89	REFRIGERATED	1	11	20

```
# copy and drop the original data to save memory
basket_data = basket_data_raw.copy()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

```
#break down by store level
basket_data['Banner'].unique()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
array(['Akins Natural Foods', 'Earth Origins Market',
       'Chamberlins Natural Foods'], dtype=object)
```

```
# break down entities by store names to come up with 3 tables
basket_data_Akins = basket_data[basket_data['Banner'] == 'Akins Natural Foods']
basket_data_Earth = basket_data[basket_data['Banner'] == 'Earth Origins Market']
basket_data_Chamberlins = basket_data[basket_data['Banner'] == 'Chamberlins Natural Foods']
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

```
basket_data_Akins.head(5)
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

	Customer.Total.Proper	Transaction.End.Date.Proper	Quantity.Sold	Base.Price	Selling.Price	Department Name Transaction	month	day	ye
0	3.99	1/14/22	1.00	3.99	3.99	REFRIGERATED	1	14	2(
1	6.99	1/14/22	0.27	6.99	1.89	BULK	1	14	2(
2	6.99	1/14/22	0.27	6.99	1.89	BULK	1	14	2(
3	6.99	1/14/22	0.27	6.99	1.89	BULK	1	14	2(
92	8.99	1/3/22	1.00	8.99	8.99	MADE-TO-ORDER DELI	1	3	2(

```
basket_data_Earth.head(5)
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

	Customer.Total.Proper	Transaction.End.Date.Proper	Quantity.Sold	Base.Price	Selling.Price	Department Name Transaction	month	day	yea
4	9.99	1/11/22	1.19	9.99	11.89	REFRIGERATED	1	11	202
5	9.99	1/11/22	1.19	9.99	11.89	REFRIGERATED	1	11	202
6	4.29	1/11/22	1.00	4.29	4.29	PRODUCE	1	11	202
7	17.99	1/11/22	1.00	17.99	17.99	SUPPLEMENTS	1	11	202
8	2.39	1/11/22	1.00	2.39	2.39	PRODUCE	1	11	202

```
basket_data_Chamberlins.head(5)
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

	Customer.Total.Proper	Transaction.End.Date.Proper	Quantity.Sold	Base.Price	Selling.Price	Department Name Transaction	month	day	y
91	6.29	1/22/22	1.00	6.29	6.29	SUPPLEMENTS	1	22	2
278	12.99	1/24/22	1.00	12.99	12.99	SUPPLEMENTS	1	24	2
279	35.99	1/24/22	1.00	35.99	35.99	PERSONAL CARE	1	24	2
280	12.99	1/3/22	0.03	12.99	0.39	BULK	1	3	2
281	12.99	1/3/22	0.03	12.99	0.39	BULK	1	3	2

EDA for Chamberlins Natural Food

```
# break down sales by annual to derive annual sales for Charmberlins
sales_Chamberlins_2021 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2021].groupby(['Category','Subcategory']).ag
sales_Chamberlins_2022 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2022].groupby(['Category','Subcategory']).ag
sales_Chamberlins_2023 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2023].groupby(['Category','Subcategory']).ag
```

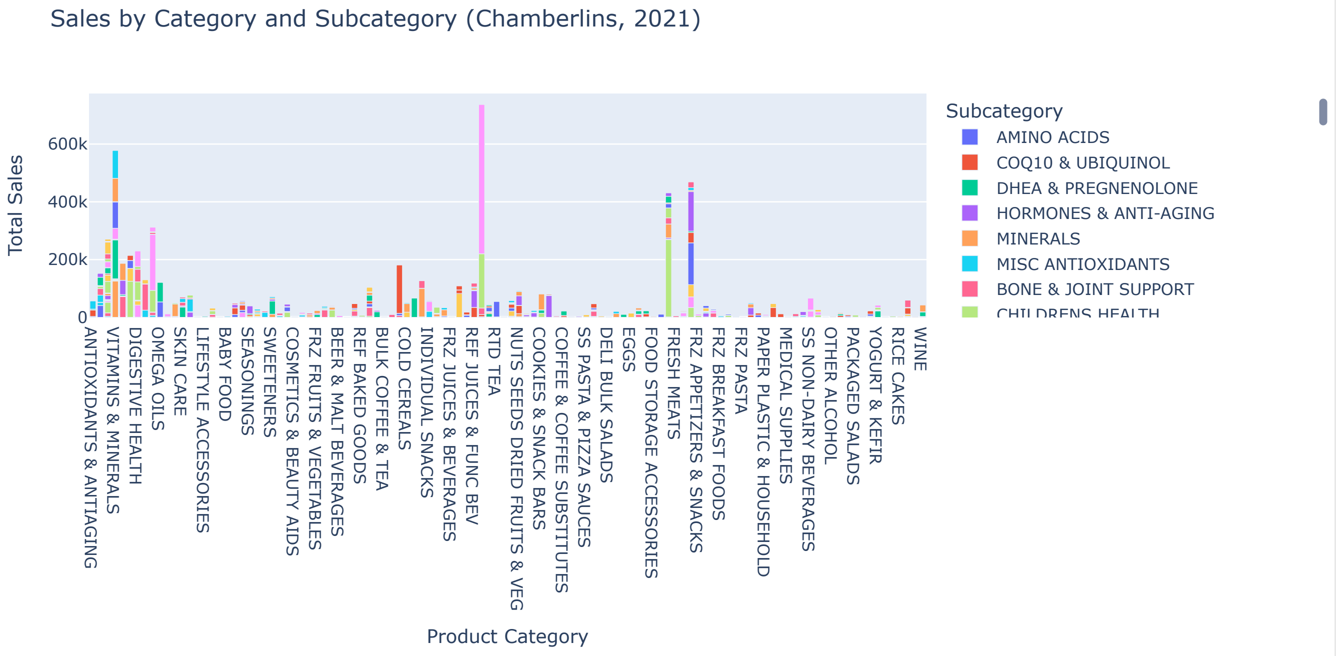
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

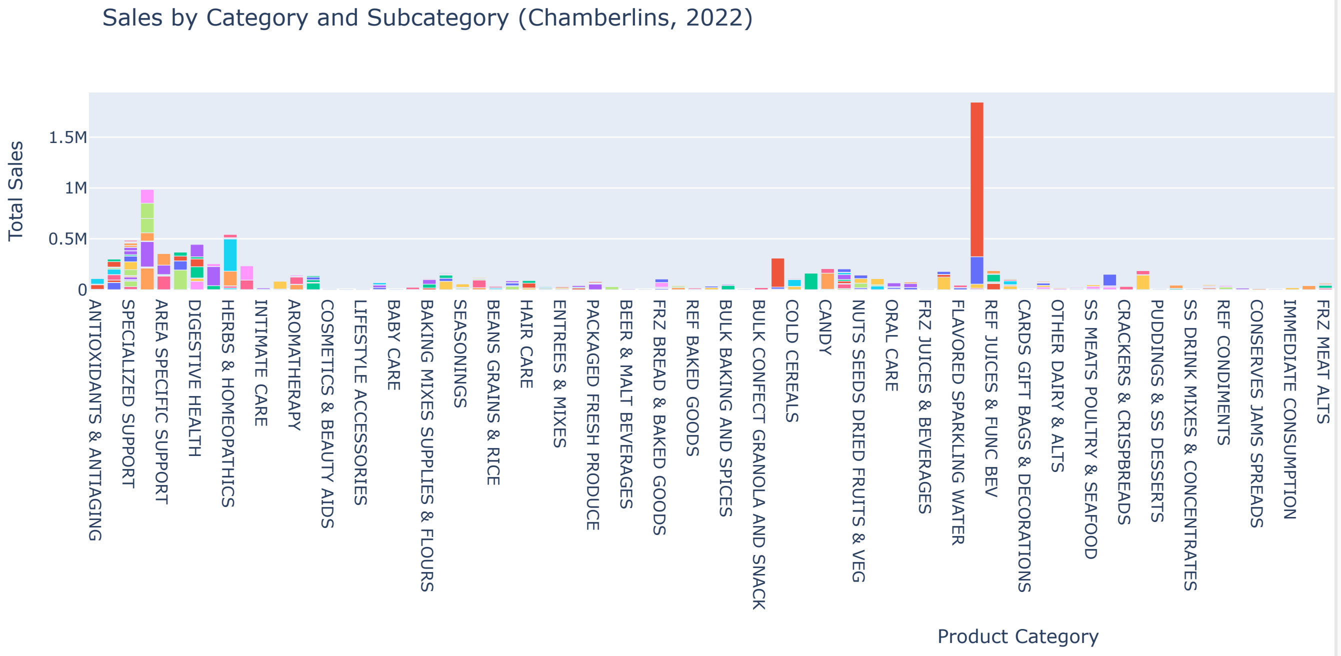
```
#reset the index of each subtable
sales_Chamberlins_2021 = sales_Chamberlins_2021.reset_index()
sales_Chamberlins_2022 = sales_Chamberlins_2022.reset_index()
sales_Chamberlins_2023 = sales_Chamberlins_2023.reset_index()
```

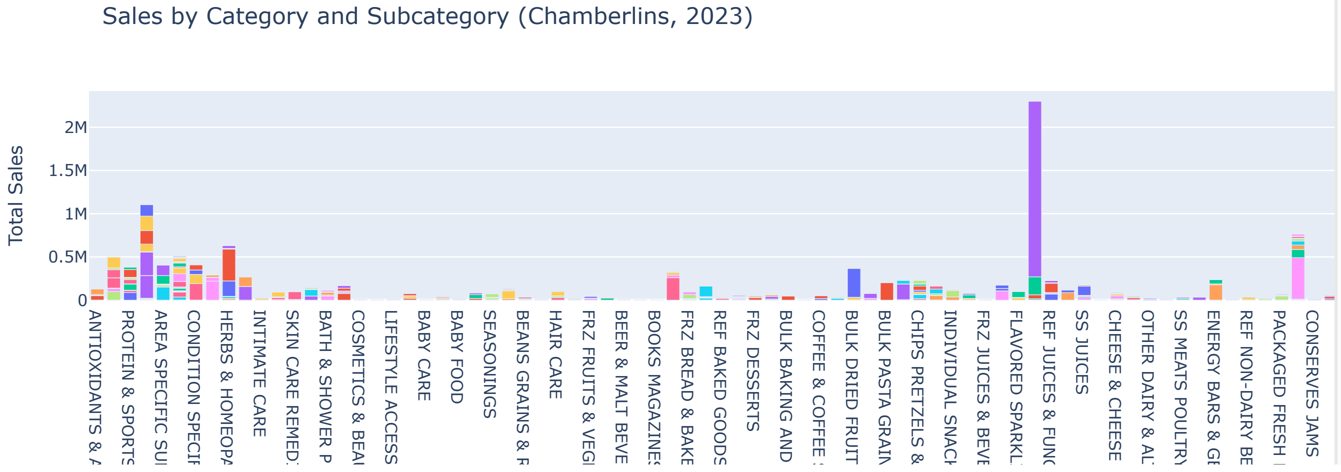
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `tra
  and should_run_async(code)
```

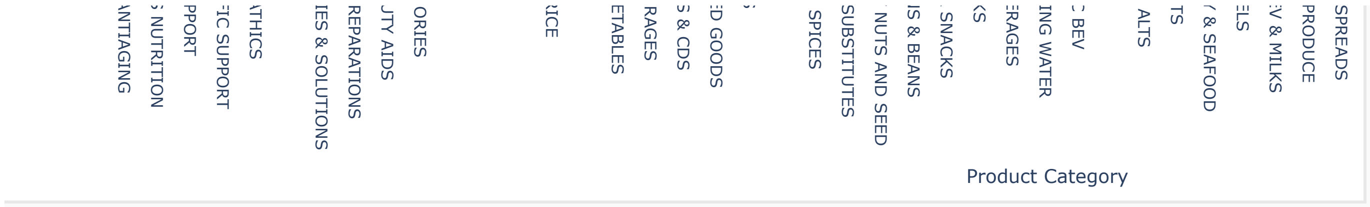
```
# Create the bar chart using Plotly Express to break down the sales attributed form each category /subcateogry

fig1 = px.bar(sales_Chamberlins_2021,
             x='Category',
             y='Sales',
             color='Subcategory',  # Color bars by subcategory
             title='Sales by Category and Subcategory (Chamberlins, 2021)',
             labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
             hover_data=['Subcategory', 'Sales'])

fig2 = px.bar(sales_Chamberlins_2022,
             x='Category',
             y='Sales',
             color='Subcategory',  # Color bars by subcategory
             title='Sales by Category and Subcategory (Chamberlins, 2022)',
             labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
             hover_data=['Subcategory', 'Sales'])

fig3 = px.bar(sales_Chamberlins_2023,
             x='Category',
             y='Sales',
                 color='Subcategory',  # Color bars by subcategory
                 title='Sales by Category and Subcategory (Chamberlins, 2023)',
                 labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
                 hover_data=['Subcategory', 'Sales'])

fig1.show()
fig2.show()
fig3.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `
  and should_run_async(code)
```









From an aggregate perspective, packaged water leads the total sales metric by generating the most revenues on Subcategory level across all years of 2021-2023, along the lines of Water generating the most revenues on the Category level. Therefore, we can clearly conclude that on the macro level, the largest share of profitability lies in the Packaged Water Subcategory. However, oen must be aware that the aggegate sales on packaged water might not be a good metrics for measuring regional/seasonal sales and might have very limitive potential of sale growth. In order to durther valdiate its profitability, we need to break down metrics by adding more control variables.

```
mon_sales_Chamberlins_water_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2021) & (basket_data_Chamberlins[
mon_sales_Chamberlins_water_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins[
mon_sales_Chamberlins_water_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2023) & (basket_data_Chamberlins[
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
mon_sales_Chamberlins_water_2021
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

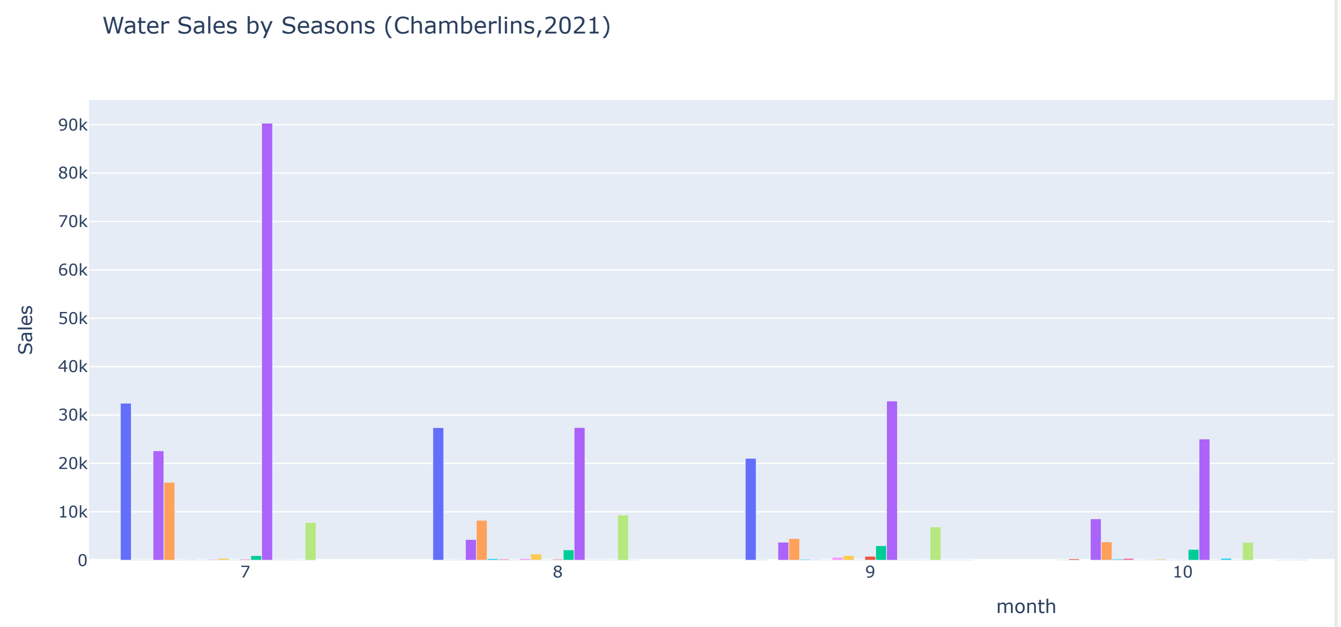
	month	Reciept Alias	Sales
0	7	ALKALINE IONIZED WATER	32440.20
1	7	ARTESIAN STILL WATER IN GLASS	76.56
2	7	ARTESIAN WATER 330 ML	29.16
3	7	HEALTHY EDGE WATER ALKALINE 9+	22605.52
4	7	HEALTHY EDGE WATER PURIFIED	16094.94
...	...	...	...
112	12	WATER ARTESIAN	229.85
113	12	WATER CHRISTMAS	6129.61
114	12	WATER SPRING ALUMINUM	16.14
115	12	WATER STILL	9.95
116	12	WATER ULTRA PURE	6897.28

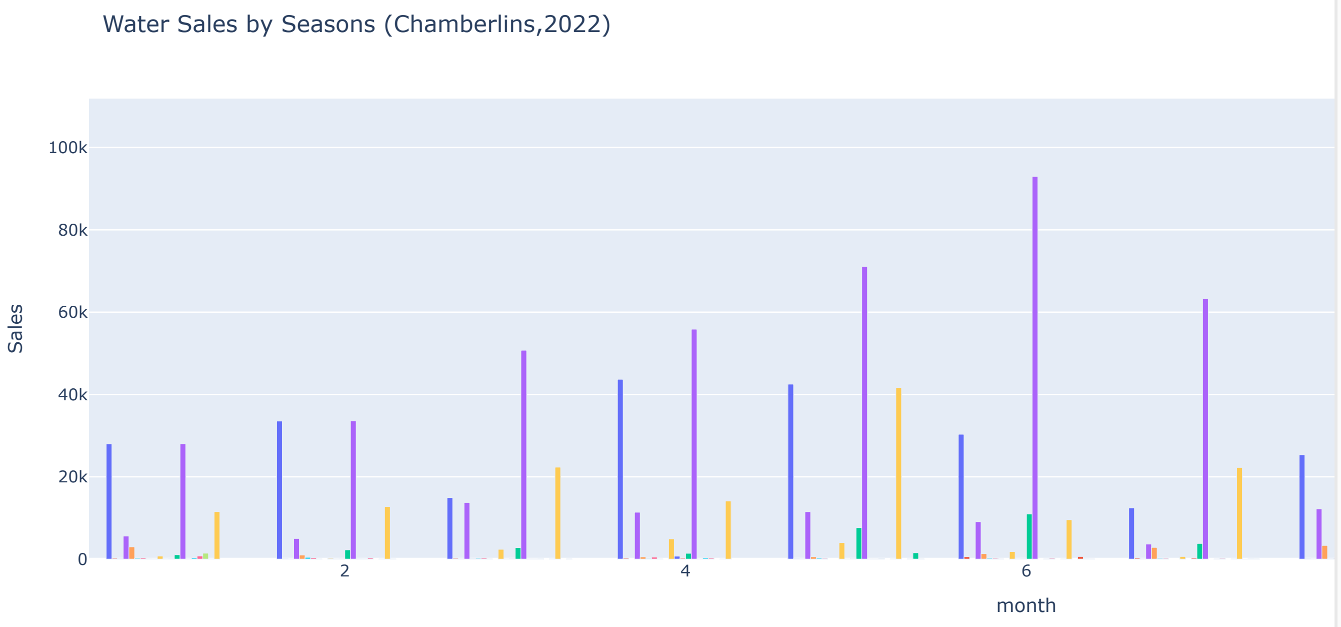
117 rows × 3 columns

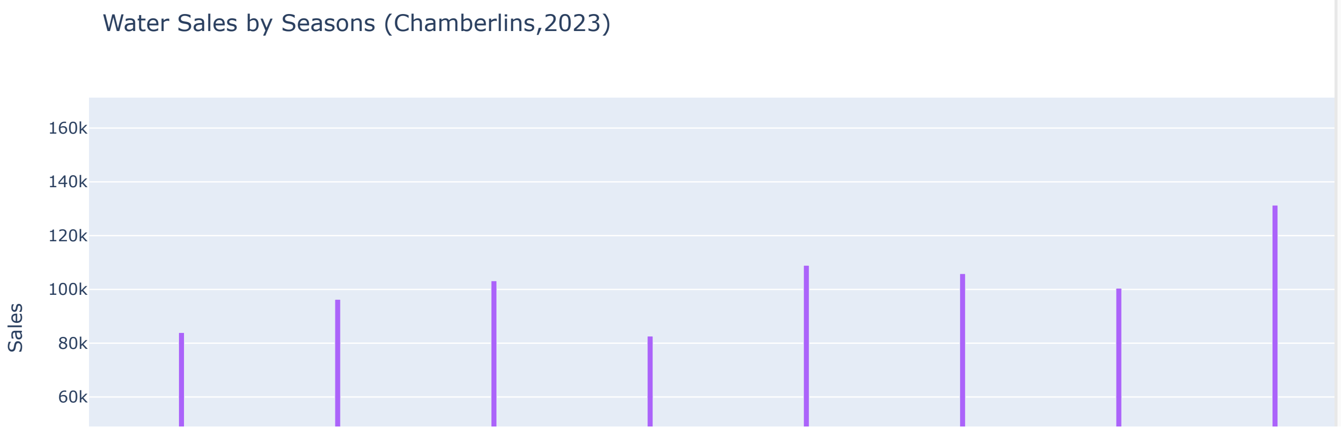
```
fig1 = px.bar(mon_sales_Chamberlins_water_2021,
             x = 'month',
             y ='Sales',
             title ='Water Sales by Seasons (Chamberlins,2021)',
             color = 'Reciept Alias',
             barmode='group'
            )

fig1.show()

fig2 = px.bar(mon_sales_Chamberlins_water_2022,
             x = 'month',
             y ='Sales',
             title ='Water Sales by Seasons (Chamberlins,2022)',
             color = 'Reciept Alias',
             barmode='group'
            )
```

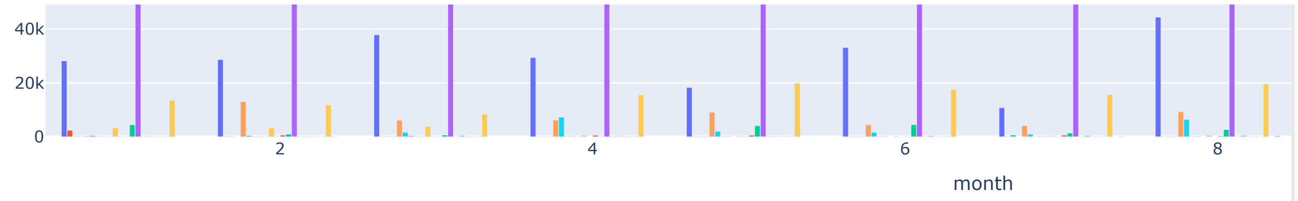
```
fig2.show()

fig3 = px.bar(mon_sales_Chamberlins_water_2023,
             x = 'month',
             y ='Sales',
             title ='Water Sales by Seasons (Chamberlins,2023)',
             color = 'Reciept Alias',
             barmode='group')

fig3.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```









Based on the subcategory level granualrity, in which we solely focus on packaged water subcategory, the Spring Water Class would consistently predominate the largest portions of sales across all seasons with sales ramped up during the 3rd and 4th quarters, which might easily misled to the conclusion that we should allcoate more marketing resource toward the Spring Water Class.

Yet that is not the case when we use from sales growth potential as the metric. The reason is by the presumption that the poriton of sales form the Spring Water Class will remain stagnated and resilient regardless of the marketing effort. That means from the global optimizaiton purpose, we should rather devote our markeitng resource into relatively low sold items that has higher sales potentials due to price elasticity, or low brand awareness, etc.

Speaking of this, when we look back into the 2021 sales, there is one uptick in sales growth for Alkaline Ionized Water in December that made it even surpass the sales of Spring Water. When I further looked back at the event calendars, I found that there exists flyer promotions for discounted Alkaline Ionized Water, which makes the price elasticity favorable towards the customer sides. Again, the same patterns happened for Ultra Pure Water during year 2022, which significantly boosted its sales growth higher among the other 2 years.

Therefore, amrkeitng campaigns should be ideally assigned towards those second-placae substitudes as they hold among the highest growth potentials (by sugesstion, regional sensitivity testing should be run to evaluate the impact).

identify seasonalities in patterns

```
mon_sales_Chamberlins_2021 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2021].groupby(['month']).agg({'Sales': "
mon_sales_Chamberlins_2022 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2022].groupby(['month']).agg({'Sales': "
mon_sales_Chamberlins_2023 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2023].groupby(['month']).agg({'Sales':"s
```

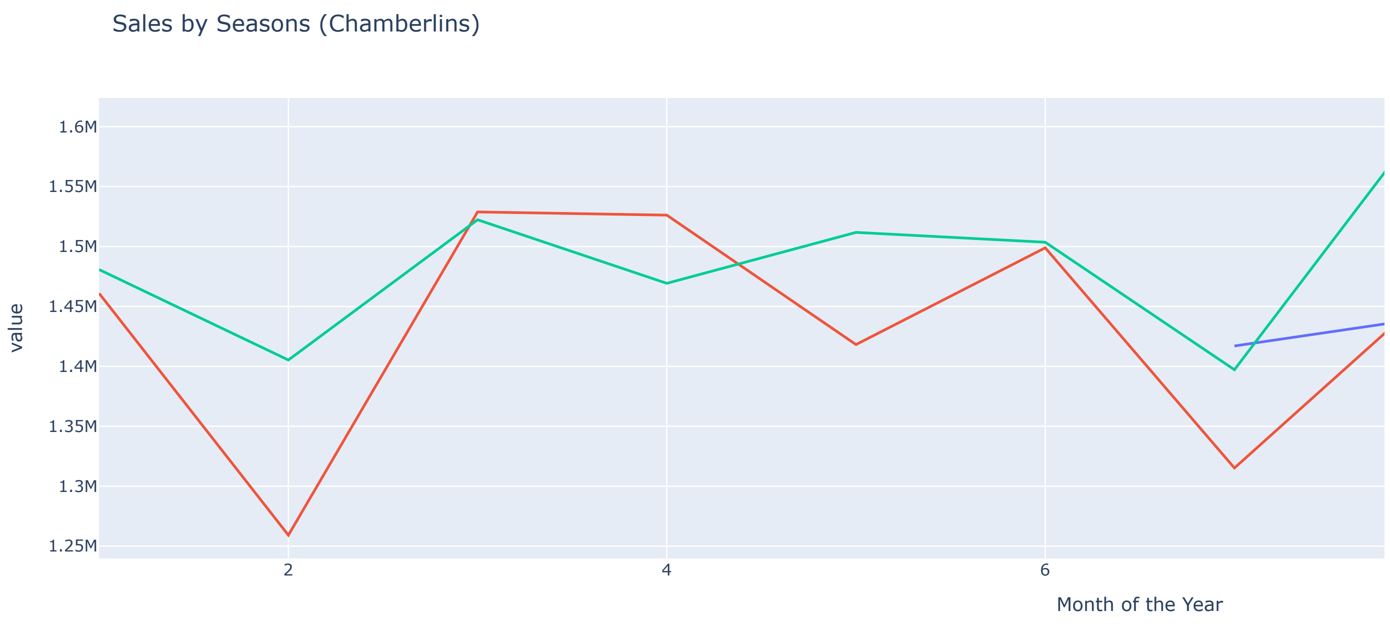
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
mon_sales_Chamberlins = mon_sales_Chamberlins_2021.merge(mon_sales_Chamberlins_2022, on='month',how='outer',left_index=False,rig
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
fig = px.line(mon_sales_Chamberlins,
             x='month',
             y =['2021 Sales','2022 Sales','2023 Sales'],
             title = 'Sales by Seasons (Chamberlins)',
             labels = {'Sales': 'Total Sales', 'month': 'Month of the Year'}
             )
fig.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```



When look at sales on monthly level, we can clearly indentify a seasonality pattern that March and August tend to have th most sales generated through the year. The sales stagnated from year 2021 to year 2022 but ramped rapidly in eyar 2023. This interpretation however are partailly incomplete, as the December sales being missed from year 2022 and the data being not recorded prior to July from year 2021.

Sales by loyalty tiers

```
loyal_sales_Chamberlins_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2021) & (basket_data_Chamberlins['Loy
loyal_sales_Chamberlins_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins['Loy
loyal_sales_Chamberlins_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2023) & (basket_data_Chamberlins['Loy
```

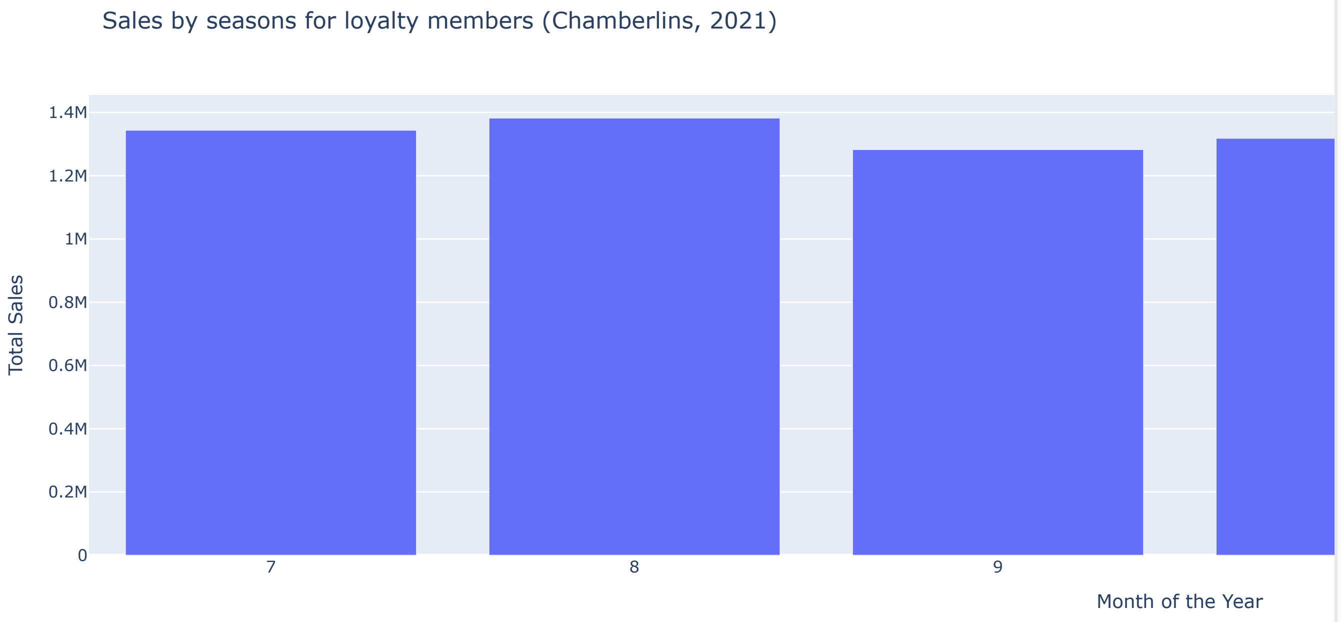
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

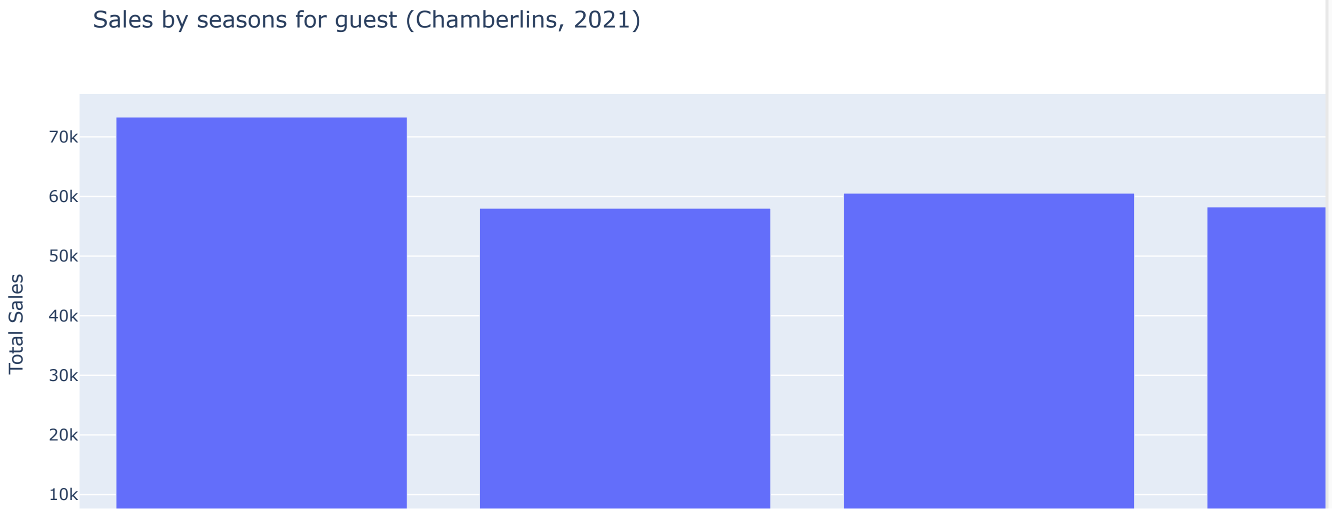
```
guest_sales_Chamberlins_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2021) & (basket_data_Chamberlins['Loy
guest_sales_Chamberlins_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins['Loy
guest_sales_Chamberlins_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2023) & (basket_data_Chamberlins['Loy
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
fig = px.bar(loyal_sales_Chamberlins_2021,
             x='month',
             y= 'Sales',
             title='Sales by seasons for loyalty members (Chamberlins, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig2 = px.bar(guest_sales_Chamberlins_2021,
             x='month',
             y= 'Sales',
             title='Sales by seasons for guest (Chamberlins, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig.show()
fig2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```

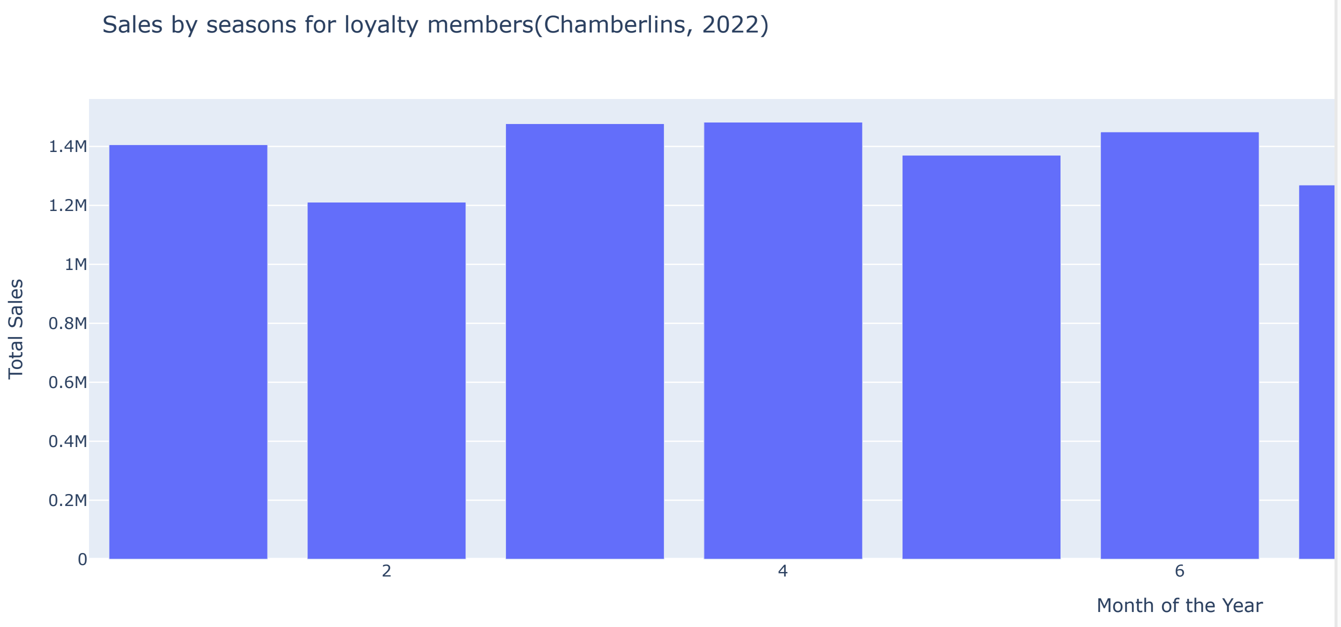


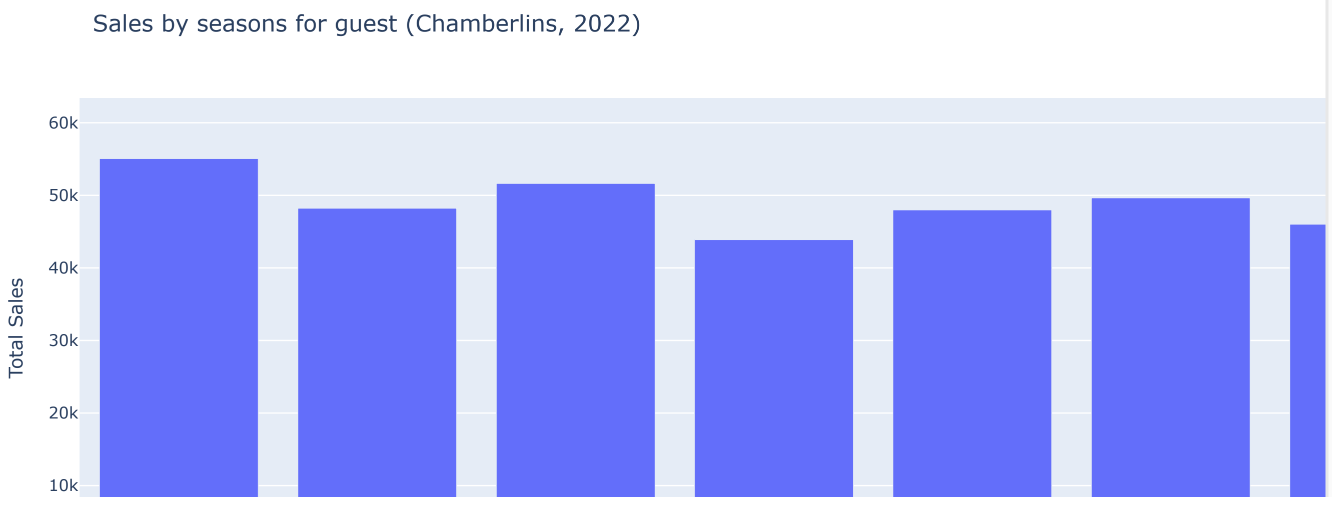


In year 2021, sales generated at steadfast speed from the loyalty members, whereas the sales from guest slightly decreased over time.

```
fig = px.bar(loyal_sales_Chamberlins_2022,
             x='month',
             y= 'Sales',
             title='Sales by seasons for loyalty members(Chamberlins, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig_2 = px.bar(guest_sales_Chamberlins_2022,
             x='month',
             y= 'Sales',
             title='Sales by seasons for guest (Chamberlins, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig.show()
fig_2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```

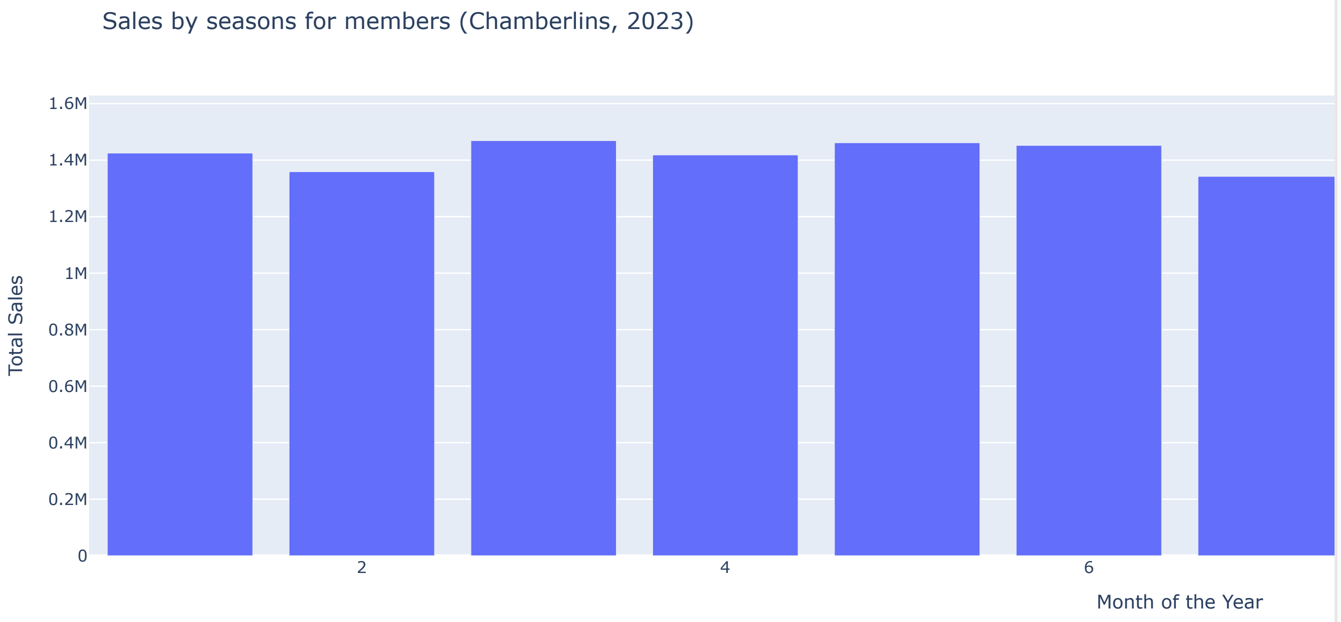


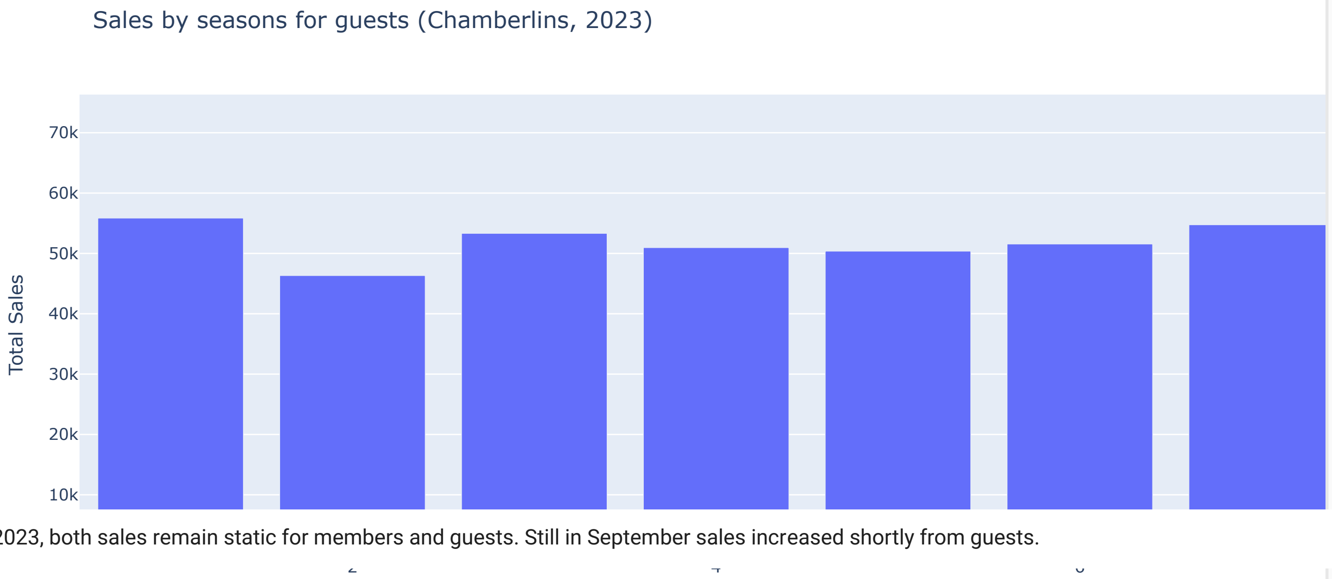


In year 2022, the sales form loyalty memebers remain static over the year.whereas the sales from guest has a slight uptick in September and decline in Octomber.

```
fig = px.bar(loyal_sales_Chamberlins_2023,
             x='month',
             y= 'Sales',
             title='Sales by seasons for members (Chamberlins, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig_2 = px.bar(guest_sales_Chamberlins_2023,
             x='month',
             y= 'Sales',
             title='Sales by seasons for guests (Chamberlins, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig.show()
fig_2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```





In year 2023, both sales remain static for members and guests. Still in September sales increased shortly from guests.

```
mon_sales_Chamberlins_water_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2021  ) & (basket_data_Chamberlir
```

Top sold item for each tier

```
item_top_5 = basket_data_Chamberlins.groupby(['Reciept Alias', 'Loyalty Customer?'])['Sales'].sum().reset_index()
loyal_item_top_5= item_top_5[item_top_5['Loyalty Customer?'] == 1].sort_values(by='Sales', ascending=False).head(5)
guest_item_top_5 = item_top_5[item_top_5['Loyalty Customer?'] == 0].sort_values(by='Sales', ascending=False).head(5)
print(loyal_item_top_5)
print(guest_item_top_5)
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`

                Reciept Alias  Loyalty Customer?         Sales
36663      SPRING WATER GLASS                  1  2.325684e+06
7363           CELERY OG EACH                  1  8.949977e+05
967    ALKALINE IONIZED WATER                  1  7.966036e+05
2305     AVOCADOS HASS OG EACH                 1  5.612368e+05
42148        WATER ULTRA PURE                  1  4.301092e+05
                     Reciept Alias  Loyalty Customer?       Sales
18659  HEALTHY EDGE WATER PURIFIED                  0  52642.4700
2304         AVOCADOS HASS OG EACH                  0  31291.8492
7362                CELERY OG EACH                  0  30587.5200
36662           SPRING WATER GLASS                  0  26463.9800
36656                 SPRING WATER                  0  18542.6800
```

Both water and veggies are the top sallers for memembers and guests. The nuances lied in the brand type: For loyalty members of Chamberlin, they tend to purchase Spring Water and Celery, whereras for guest they tend to purchase Health Edge Water and Avocados more.

Sales trends monthly for members and guests for goods Spring Water Glass

```
basket_data_Chamberlins_member_SpringWater_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRING W
basket_data_Chamberlins_member_SpringWater_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRING W
basket_data_Chamberlins_member_SpringWater_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRING W
basket_data_Chamberlins_member_Celery_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'CELERY OG EAC
basket_data_Chamberlins_member_Celery_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'CELERY OG EAC
basket_data_Chamberlins_member_Celery_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'CELERY OG EAC
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
basket_data_Chamberlins_guest_SpringWater_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRING WA
basket_data_Chamberlins_guest_SpringWater_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRING WA
basket_data_Chamberlins_guest_SpringWater_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRING WA
basket_data_Chamberlins_guest_Celery_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'CELERY OG EACH
basket_data_Chamberlins_guest_Celery_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'CELERY OG EACH
basket_data_Chamberlins_guest_Celery_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'CELERY OG EACH
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

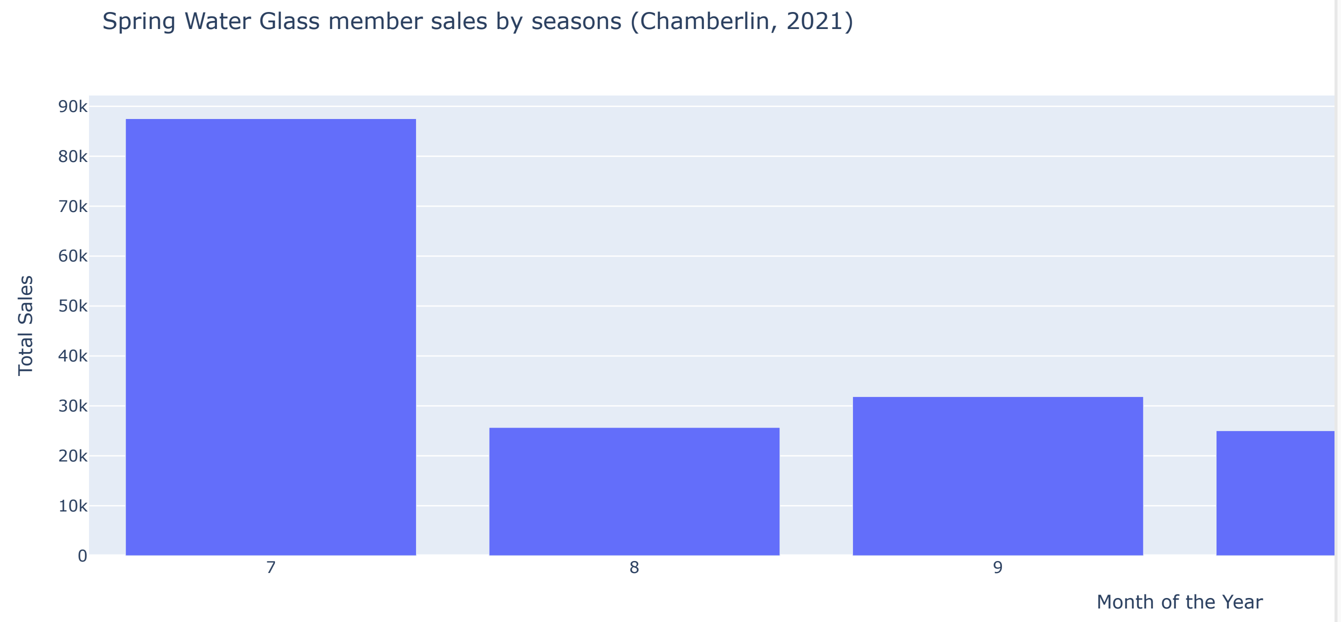
```
basket_data_Chamberlins_guest_HE_Purified_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'HEALTHY E
basket_data_Chamberlins_guest_HE_Purified_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'HEALTHY E
basket_data_Chamberlins_guest_HE_Purified_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'HEALTHY E
```

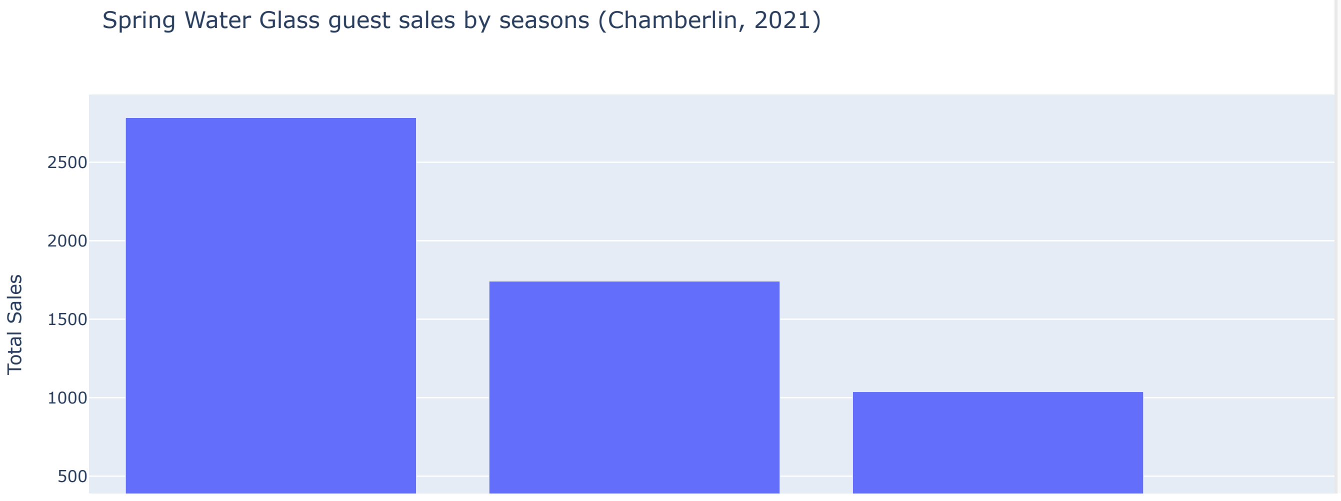
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
fig = px.bar(basket_data_Chamberlins_member_SpringWater_2021,
             x = 'month',
             y= 'Sales',
             title = 'Spring Water Glass member sales by seasons (Chamberlin, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig.show()
```

```
fig2 = px.bar(basket_data_Chamberlins_guest_SpringWater_2021,
             x = 'month',
             y= 'Sales',
             title = 'Spring Water Glass guest sales by seasons (Chamberlin, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig2.show()

fig3 = px.bar(basket_data_Chamberlins_guest_HE_Purified_2021,
             x = 'month',
             y= 'Sales',
             title = 'Healthy Edger Water Purified guest sales by seasons (Chamberlin, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig3.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```



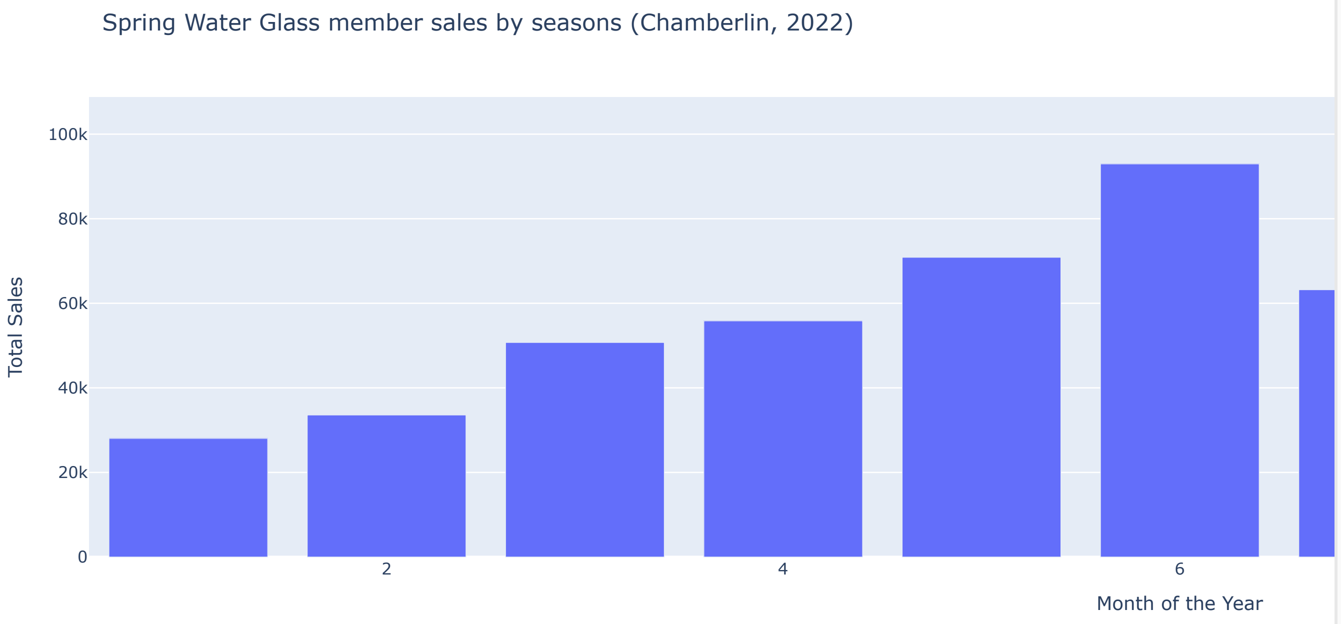


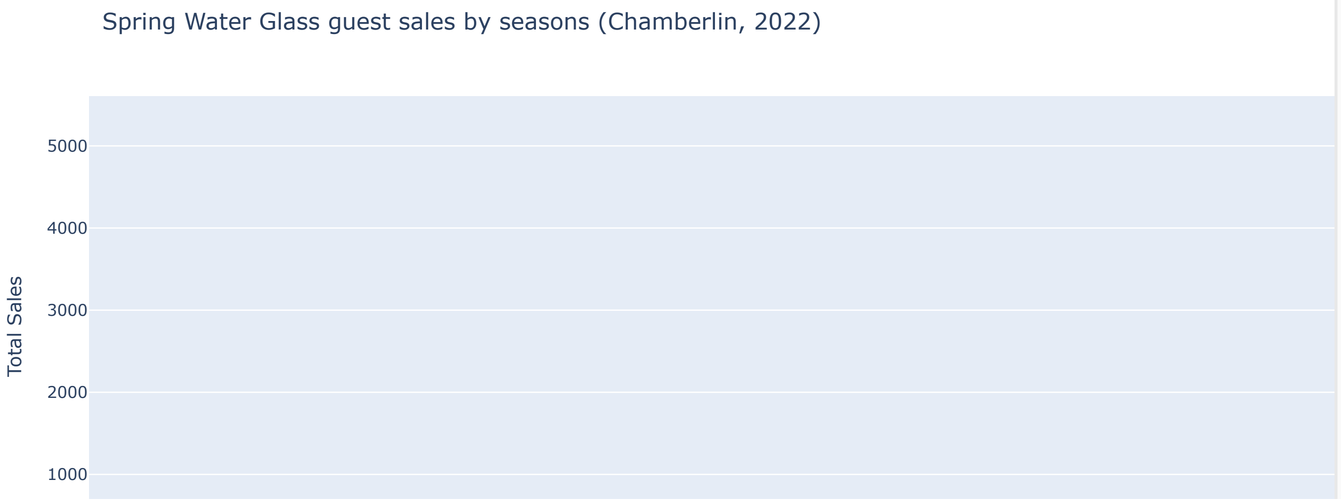
In year 2021, the sales of H.E. Purified Water significantly surpass that of the Spring Water Glass within guest group. This could mean that the pre-sale price of H.E. Purified water can be significantly lower than the Spring Glass.

```
fig = px.bar(basket_data_Chamberlins_member_SpringWater_2022,
             x = 'month',
             y= 'Sales',
             title = 'Spring Water Glass member sales by seasons (Chamberlin, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig.show()
```

```
fig2 = px.bar(basket_data_Chamberlins_guest_SpringWater_2022,
             x = 'month',
             y= 'Sales',
             title = 'Spring Water Glass guest sales by seasons (Chamberlin, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig2.show()

fig3 = px.bar(basket_data_Chamberlins_guest_HE_Purified_2022,
             x = 'month',
             y= 'Sales',
             title = 'Healthy Edger Water Purified guest sales by seasons (Chamberlin, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig3.show()
```

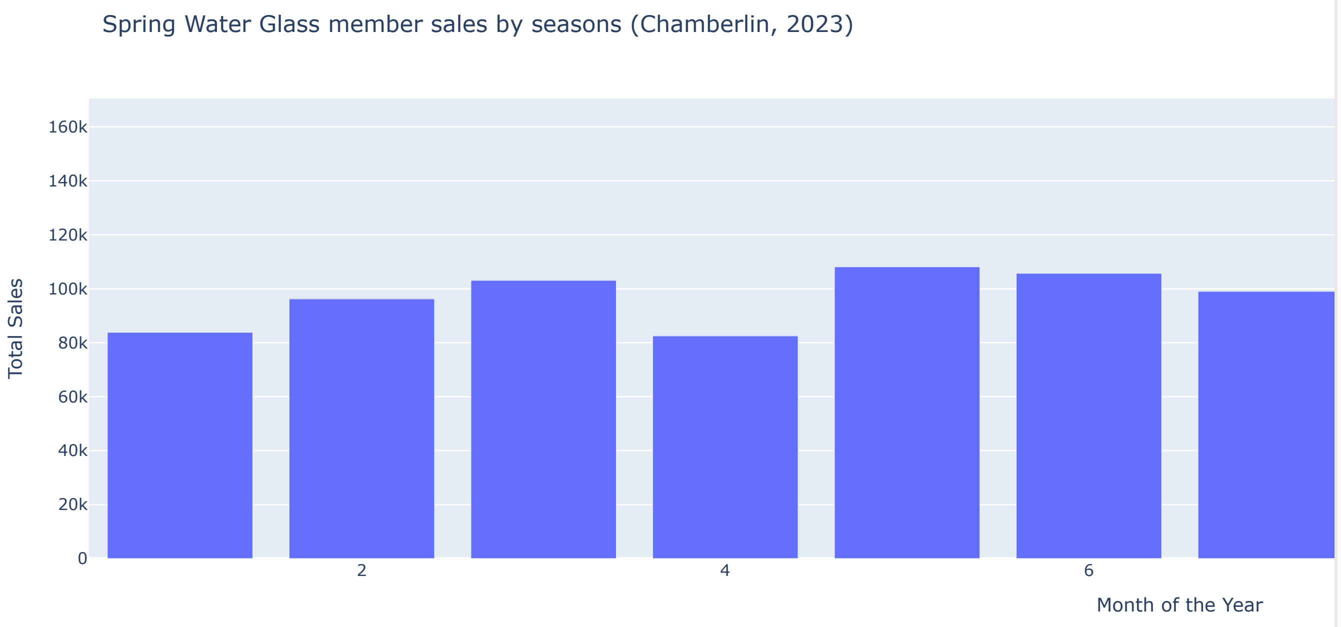
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```

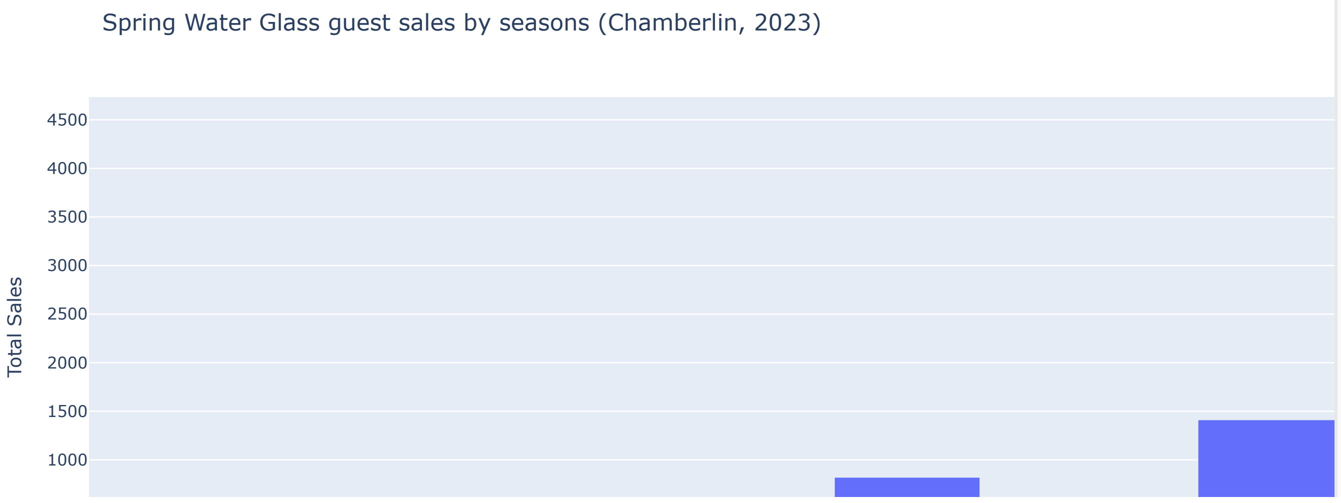




```
fig = px.bar(basket_data_Chamberlins_member_SpringWater_2023,
             x = 'month',
             y= 'Sales',
             title = 'Spring Water Glass member sales by seasons (Chamberlin, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig.show()

fig2 = px.bar(basket_data_Chamberlins_guest_SpringWater_2023,
             x = 'month',
             y= 'Sales',
             title = 'Spring Water Glass guest sales by seasons (Chamberlin, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig2.show()

fig3 = px.bar(basket_data_Chamberlins_guest_HE_Purified_2023,
             x = 'month',
             y= 'Sales',
             title = 'Healthy Edger Water Purified guest sales by seasons (Chamberlin, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig3.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```



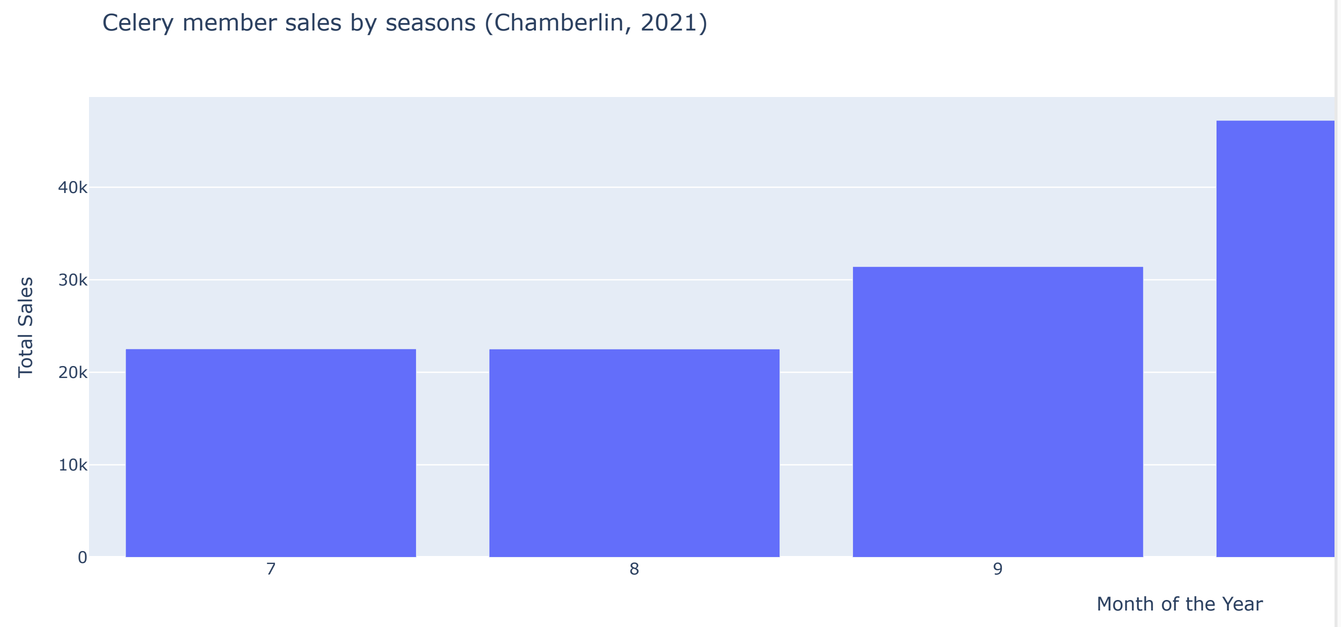


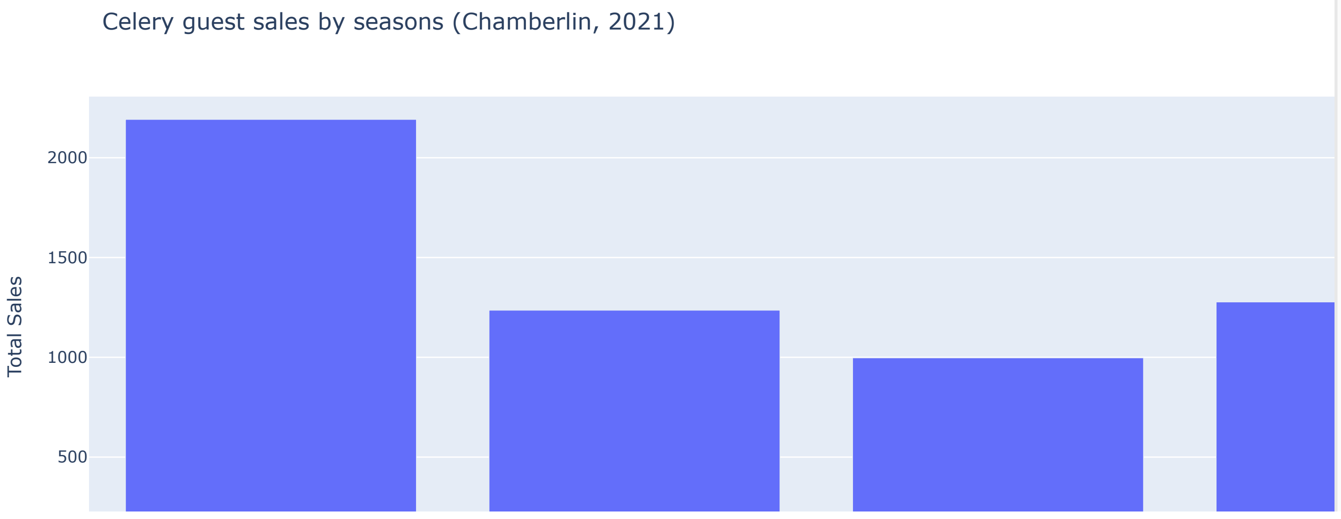
In year 2023, The correlationsales of H.E. Purified Water plummeted to zero on November and December. However, the sales of Spring Water peaked at the end of the year. THis cotnradictory might be attribtued by some kind of sales events that drop the price of Spring Water significantly lower than that of H.E. Purified Water.

```
fig = px.bar(basket_data_Chamberlins_member_Celery_2021,
            x = 'month',
            y= 'Sales',
            title = 'Celery member sales by seasons (Chamberlin, 2021)',
            labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig.show()
```

```
fig2 = px.bar(basket_data_Chamberlins_guest_Celery_2021,
             x = 'month',
             y= 'Sales',
             title = 'Celery guest sales by seasons (Chamberlin, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig2.show()
```

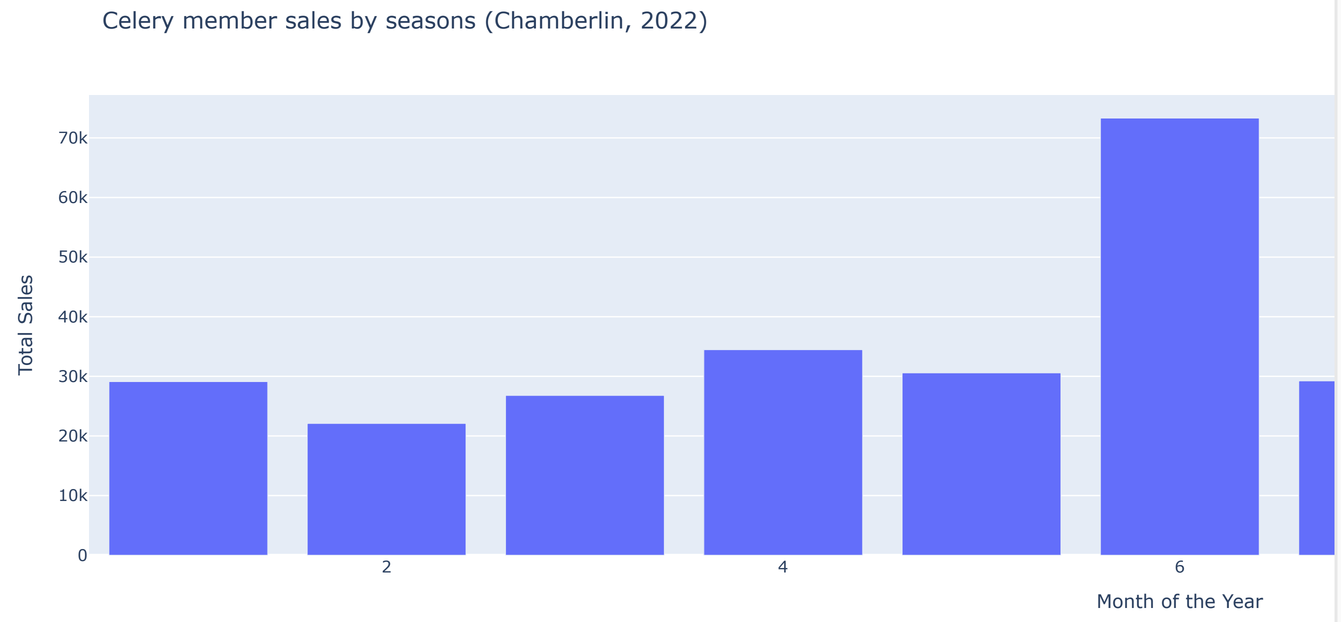
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```

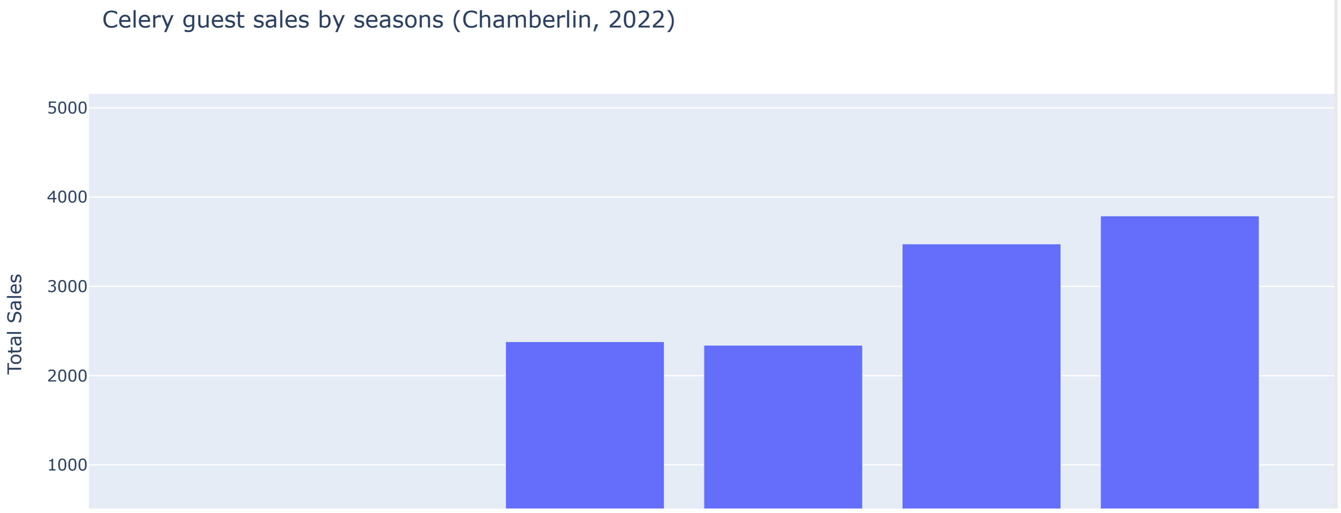




```
fig = px.bar(basket_data_Chamberlins_member_Celery_2022,
             x = 'month',
             y= 'Sales',
             title = 'Celery member sales by seasons (Chamberlin, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig.show()

fig2 = px.bar(basket_data_Chamberlins_guest_Celery_2022,
             x = 'month',
             y= 'Sales',
             title = 'Celery guest sales by seasons (Chamberlin, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig2.show()
```

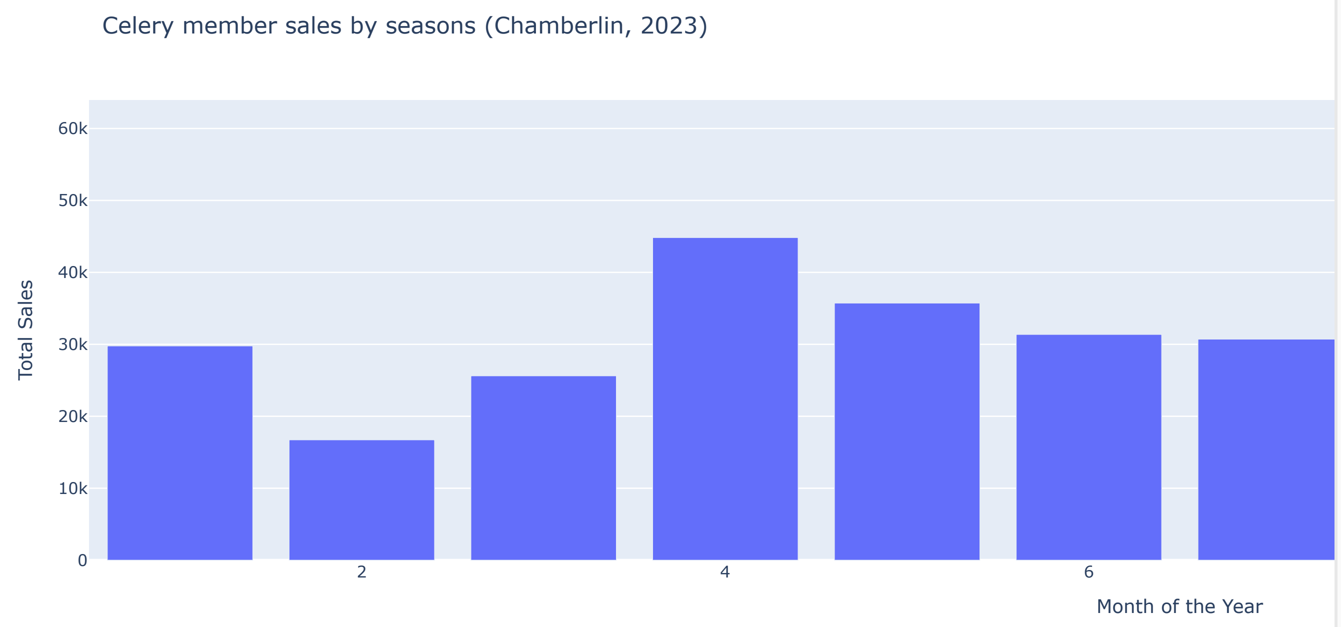
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```

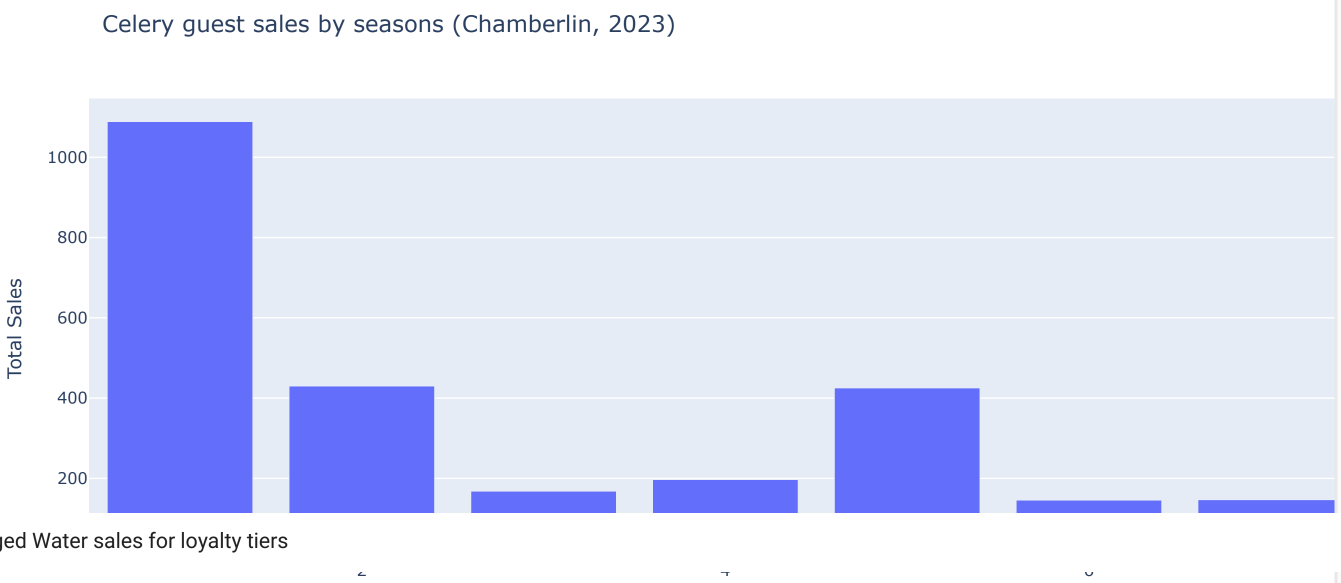




```
fig = px.bar(basket_data_Chamberlins_member_Celery_2023,
             x = 'month',
             y= 'Sales',
             title = 'Celery member sales by seasons (Chamberlin, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig.show()

fig2 = px.bar(basket_data_Chamberlins_guest_Celery_2023,
             x = 'month',
             y= 'Sales',
             title = 'Celery guest sales by seasons (Chamberlin, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```





Packaged Water sales for loyalty tiers

```
#Aggregate total sales for guest customer from 2021-2023
mon_sales_Chamberlins_guest_packaged_water_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['Loyalty Customer?'] == 0) &
mon_sales_Chamberlins_guest_packaged_water_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Loyalty Customer?'] == 0) &
mon_sales_Chamberlins_guest_packaged_water_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['Loyalty Customer?'] == 0) &
```

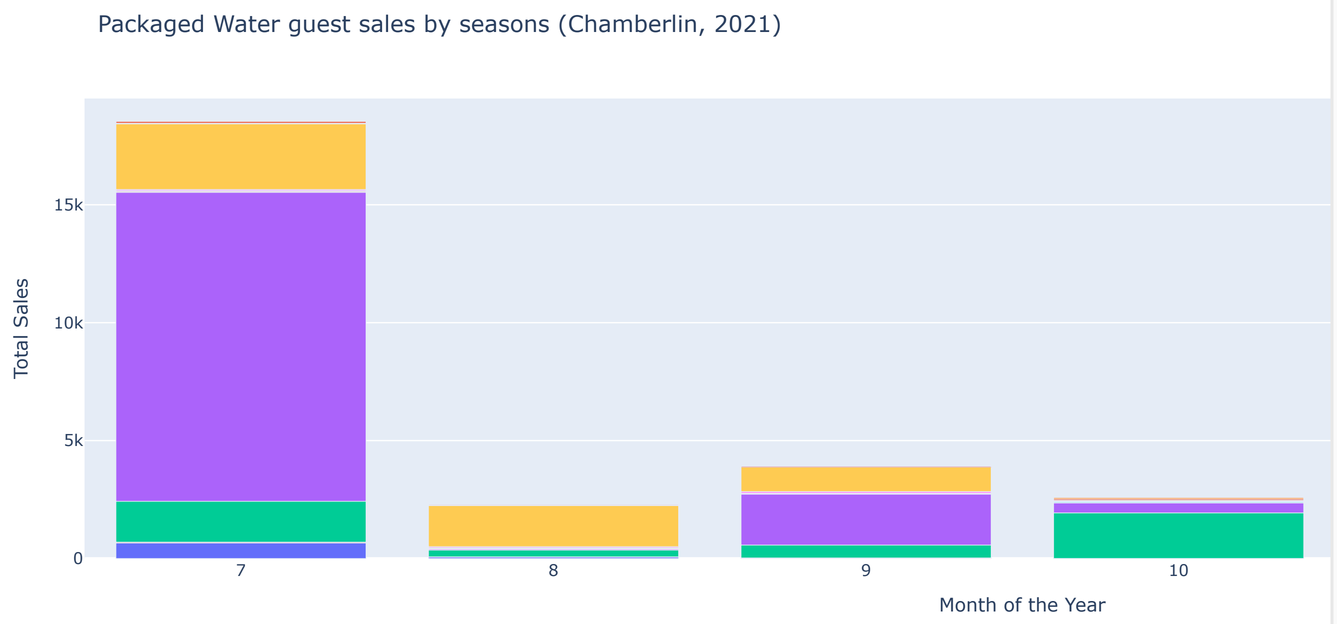
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

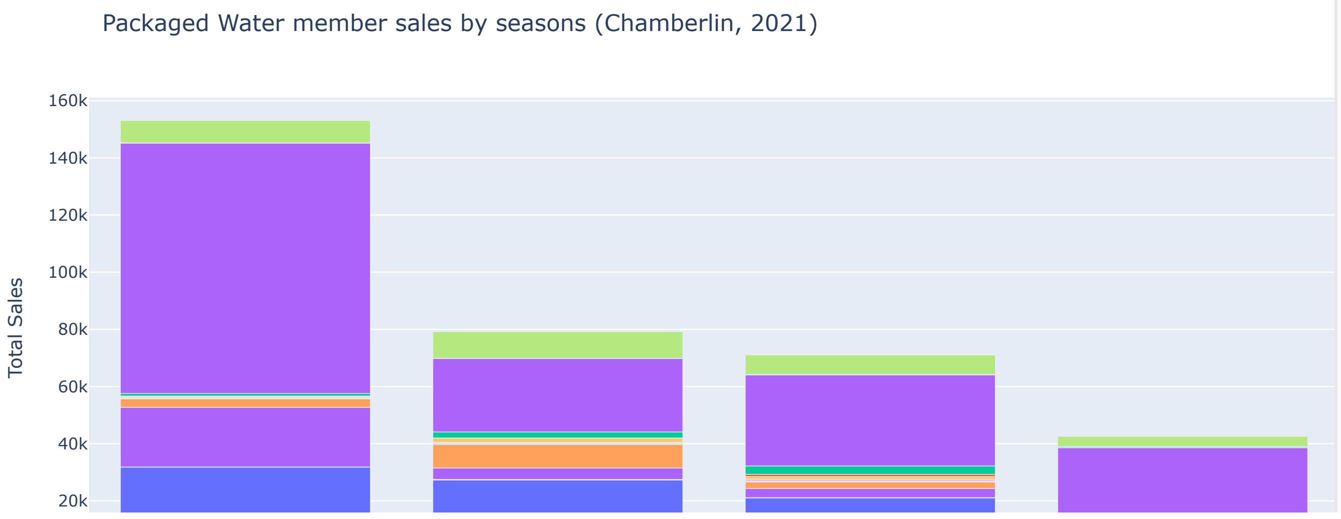
```
mon_sales_Chamberlins_member_packaged_water_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['Loyalty Customer?'] == 1) &
mon_sales_Chamberlins_member_packaged_water_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Loyalty Customer?'] == 1) &
mon_sales_Chamberlins_member_packaged_water_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['Loyalty Customer?'] == 1) &
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
fig = px.bar(mon_sales_Chamberlins_guest_packaged_water_2021,
             x = 'month',
             y= 'Sales',
             color = 'Reciept Alias',
             title = 'Packaged Water guest sales by seasons (Chamberlin, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig.show()

fig2 = px.bar(mon_sales_Chamberlins_member_packaged_water_2021,
             x = 'month',
             y= 'Sales',
             color = 'Reciept Alias',
             title = 'Packaged Water member sales by seasons (Chamberlin, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```



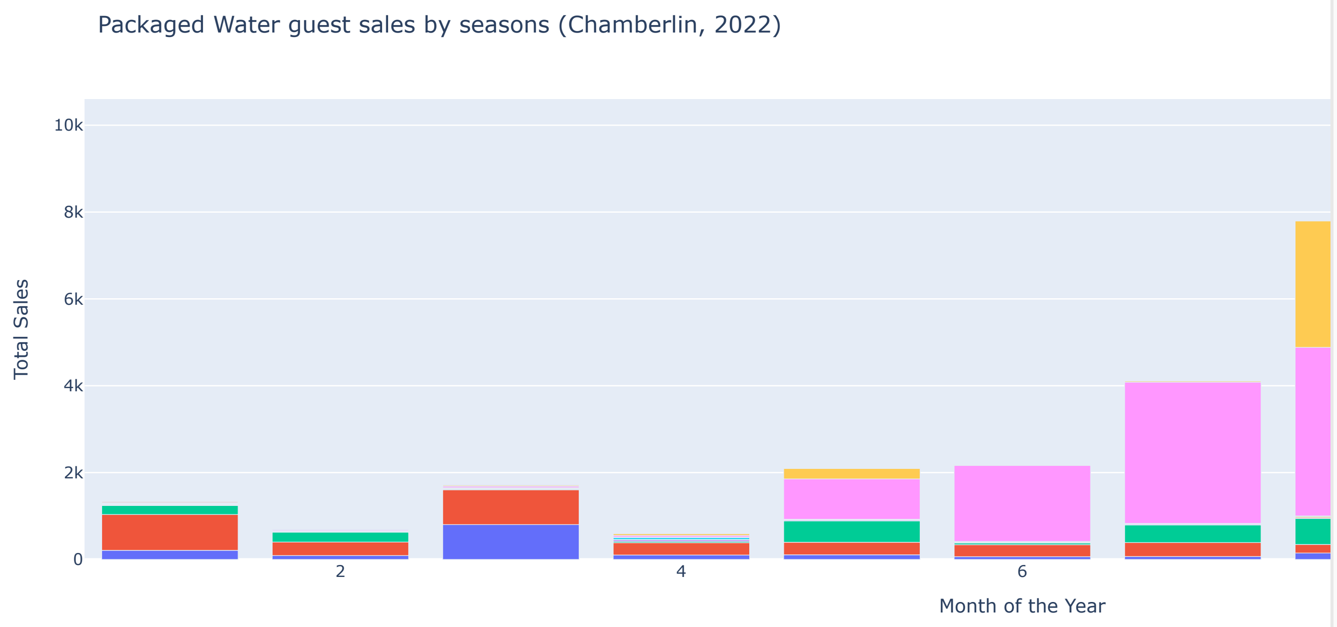


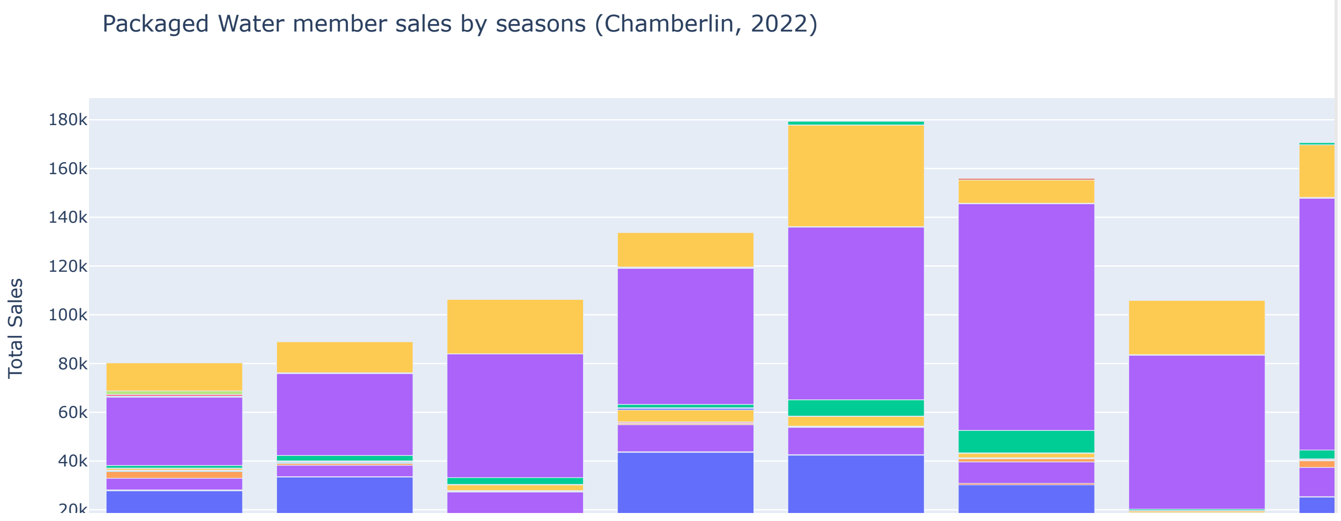
For guest customer, July's sales were mainly contributed by the Healthy Edge Purified Water, which covered the most of sales occured during the year. But the trend shift to Healthy Edge Alkatine taking over the most sold item with the following months.

For loyalty customer, Spring Water Class was always the major part of the monthly sales, with Alkatine Ionized Water taking over some portions during the July, August, Septemebr and December. Again, the sales peak in month July.

```
fig = px.bar(mon_sales_Chamberlins_guest_packaged_water_2022,
             x = 'month',
             y= 'Sales',
             color = 'Reciept Alias',
             title = 'Packaged Water guest sales by seasons (Chamberlin, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig.show()

fig2 = px.bar(mon_sales_Chamberlins_member_packaged_water_2022,
             x = 'month',
             y= 'Sales',
             color = 'Reciept Alias',
             title = 'Packaged Water member sales by seasons (Chamberlin, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```



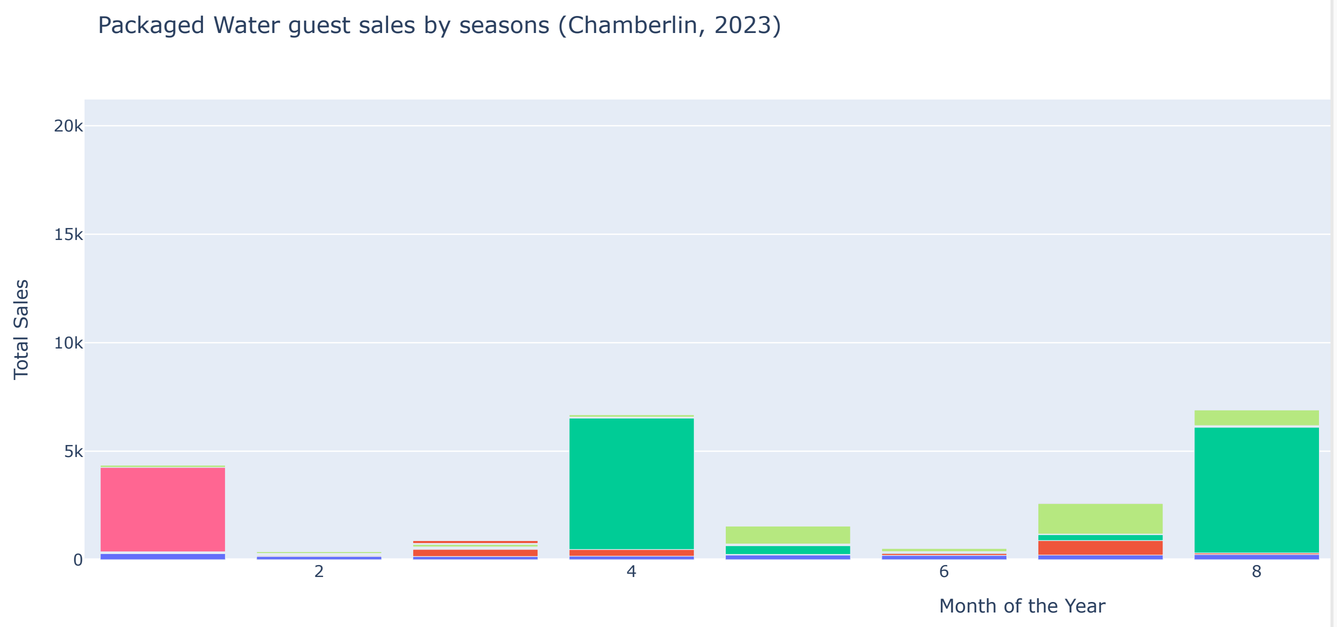


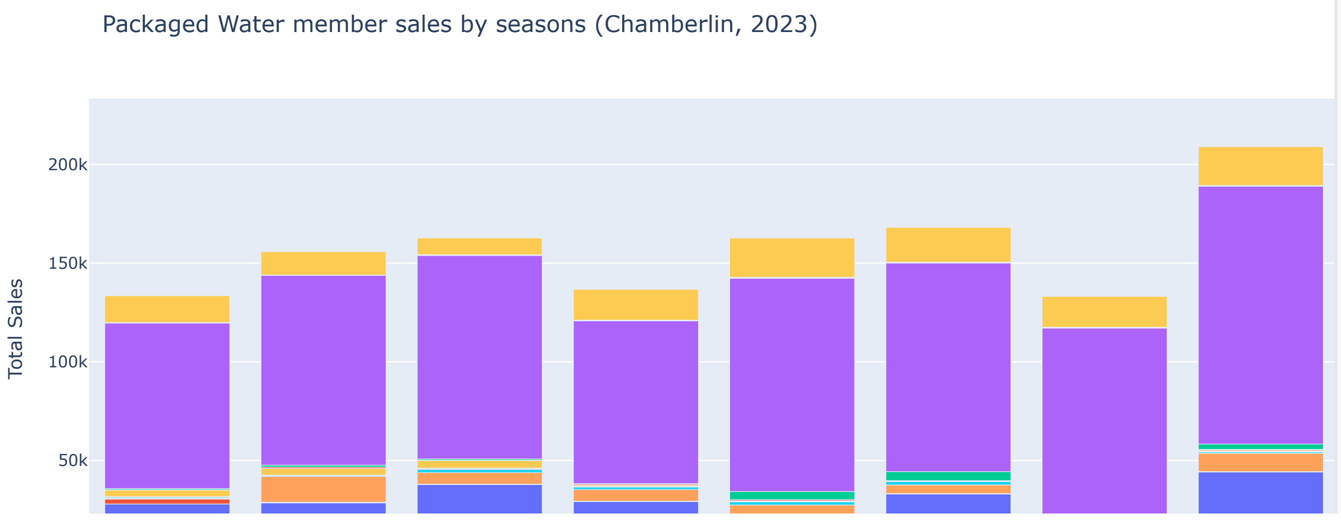
For guest customer, sales were attributed from Spring Water between May and Octomber. It is worth noticing that in September, the sales for Spring Water Glass surged so high that it became the most sold item of the year for the guest group.

For loyalty customer, the sales were attributed mainly from the Spring Water Glass and remained static over the months.

```
fig = px.bar(mon_sales_Chamberlins_guest_packaged_water_2023,
             x = 'month',
             y= 'Sales',
             color = 'Reciept Alias',
             title = 'Packaged Water guest sales by seasons (Chamberlin, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig.show()

fig2 = px.bar(mon_sales_Chamberlins_member_packaged_water_2023,
             x = 'month',
             y= 'Sales',
             color = 'Reciept Alias',
             title = 'Packaged Water member sales by seasons (Chamberlin, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'})
fig2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```





Year 2023 has strong divergency over the item preferences between the two customer groups.

For guest customer, the sales were attributed from the Healthy Edged Purified Water and the majority of sales occured during September.

For loyalty customer, the sales were attributed from the Spring Water Glass mainly as the demand remain storng and static over the year.

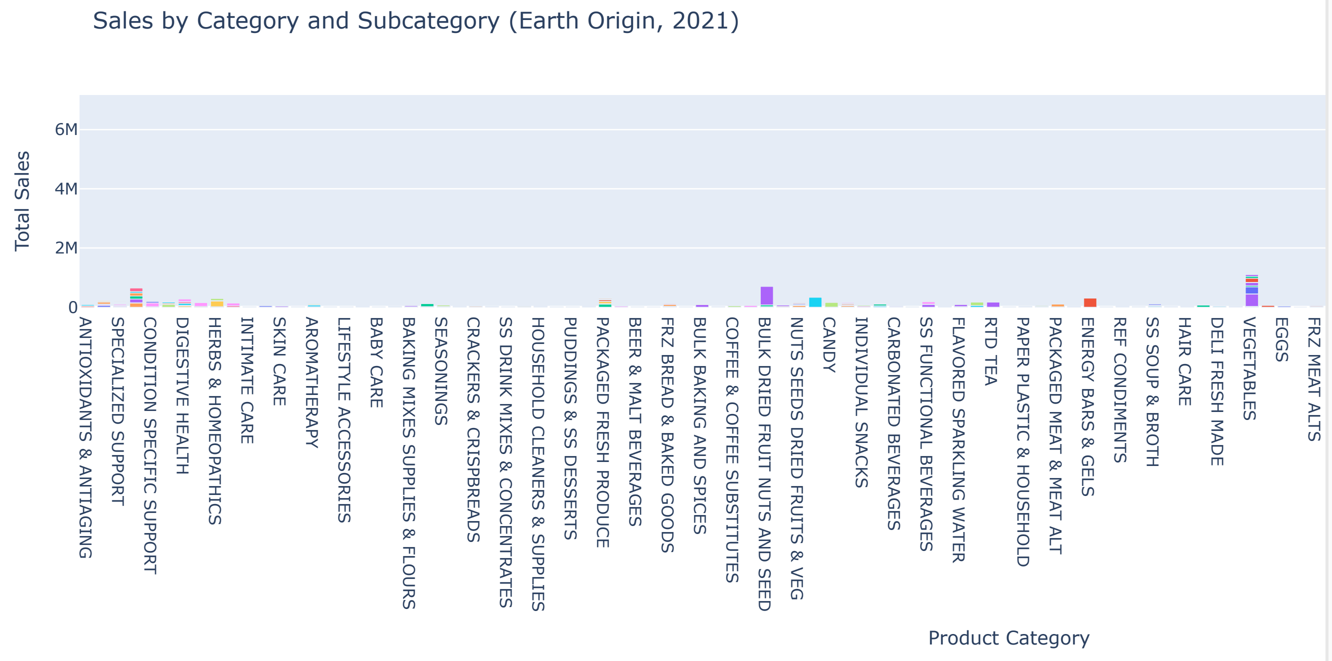
EDA for Earth Origins

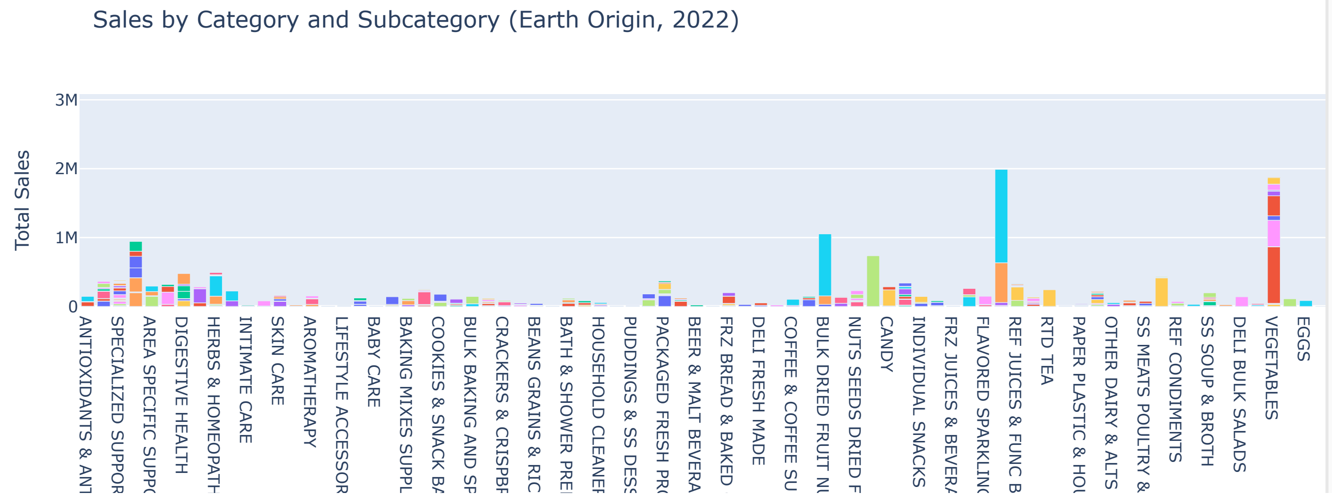
```
sales_Earth_2021 = basket_data_Earth[basket_data_Earth['year'] == 2021].groupby(['Category','Subcategory']).agg({'Sales': "sum"}
sales_Earth_2022 = basket_data_Earth[basket_data_Earth['year'] == 2022].groupby(['Category','Subcategory']).agg({'Sales': "sum"}
sales_Earth_2023 = basket_data_Earth[basket_data_Earth['year'] == 2023].groupby(['Category','Subcategory']).agg({'Sales': "sum"}
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
# Create the bar chart using Plotly Express

fig1 = px.bar(sales_Earth_2021,
             x='Category',
             y='Sales',
             color='Subcategory',  # Color bars by subcategory
             title='Sales by Category and Subcategory (Earth Origin, 2021)',
             labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
             hover_data=['Subcategory', 'Sales'])

fig2 = px.bar(sales_Earth_2022,
             x='Category',
             y='Sales',
             color='Subcategory',  # Color bars by subcategory
             title='Sales by Category and Subcategory (Earth Origin, 2022)',
             labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
             hover_data=['Subcategory', 'Sales'])

fig3 = px.bar(sales_Earth_2023,
             x='Category',
             y='Sales',
             color='Subcategory',  # Color bars by subcategory
             title='Sales by Category and Subcategory (Earth Origin, 2023)',
             labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
             hover_data=['Subcategory', 'Sales'])

fig1.show()
fig2.show()
fig3.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```





In year 2021, the most sold category is still Water, alike to the Akins store.One trend worth notcing here is that the water pedominatly lead the most portion of sales throughout the year. Things became different in year 2022, as the fruite cateogory started to dominate the sales generation. Year 2023 has the same trend with fruit being the top sold cateogry.

identify seasonalities in patterns

```
mon_sales_Earth_2021 = basket_data_Earth[basket_data_Earth['year'] == 2021].groupby(['month']).agg({'Sales': "sum"}).reset_index
mon_sales_Earth_2022 = basket_data_Earth[basket_data_Earth['year'] == 2022].groupby(['month']).agg({'Sales': "sum"}).reset_index
mon_sales_Earth_2023 = basket_data_Earth[basket_data_Earth['year'] == 2023].groupby(['month']).agg({'Sales':"sum"}).reset_index(
```

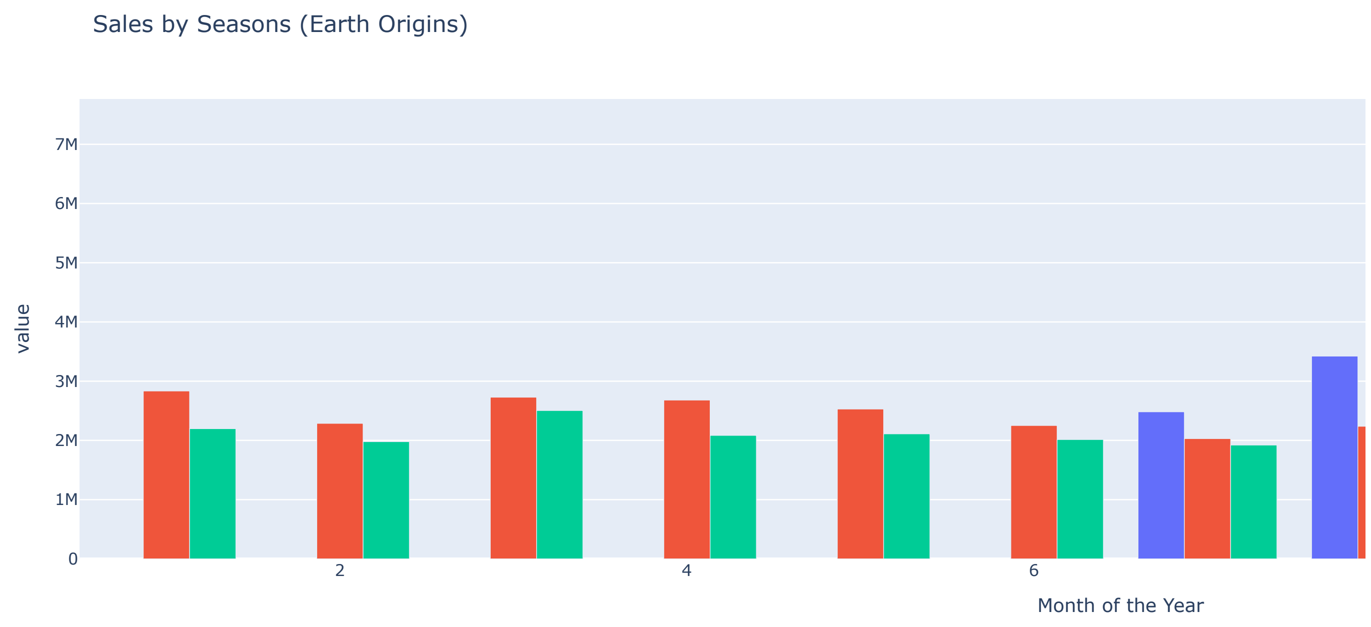
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
mon_sales_Earth = mon_sales_Earth_2021.merge(mon_sales_Earth_2022, on='month',how='outer',left_index=False,right_index=False).re
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
fig = px.bar(mon_sales_Earth,
             x='month',
             y =['2021 Sales','2022 Sales','2023 Sales'],
             title = 'Sales by Seasons (Earth Origins)',
             barmode='group',
             labels = {'Sales': 'Total Sales', 'month': 'Month of the Year'}
             )
fig.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```



Following the patterns defined from the category breakdowns, we can attribute the trend of the sales spike in September 2021 to the high demand from packaged water. This marks some abnormal sales-driver activities from the September which lead to that high sales growth (i.e. marketing campaign, superday, promotion, etc.)

Let's delve into teh specific sales pattern in September.

```
mon_sales_Earth_2021_9 = basket_data_Earth[(basket_data_Earth['year'] == 2021) & (basket_data_Earth['month'] == 9)].groupby(['mc
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

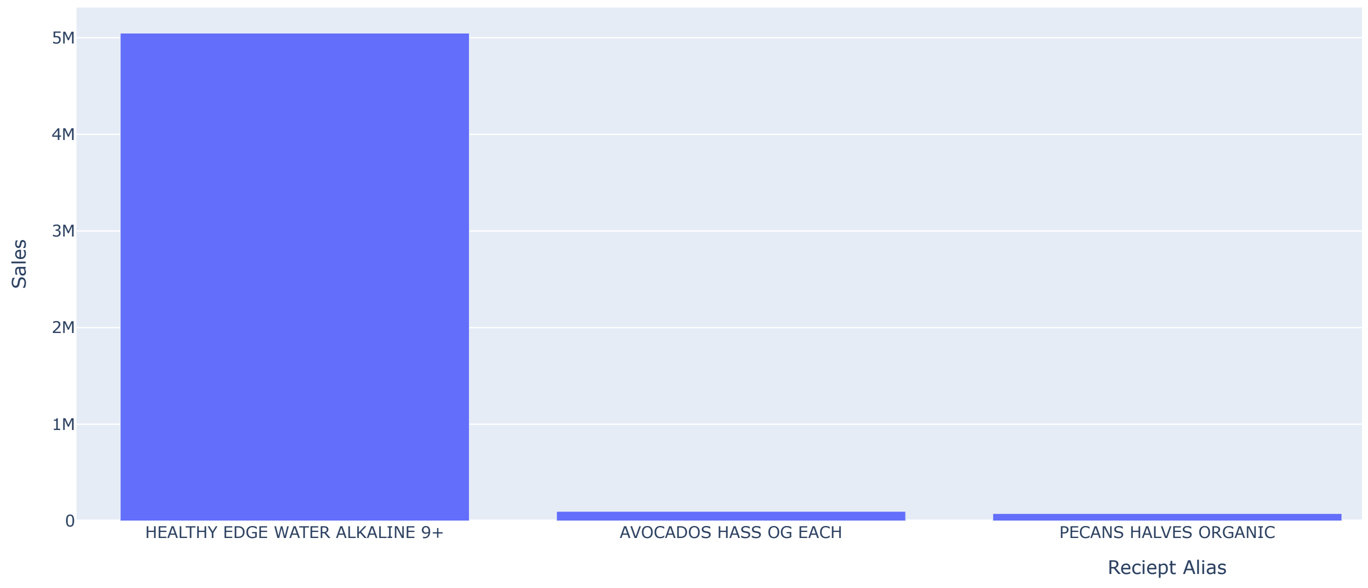
```
#Query the top sold 5 items from the month
mon_sales_Earth_2021_9
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

	month	Reciept Alias	Sales
7200	9	HEALTHY EDGE WATER ALKALINE 9+	5047331.150
902	9	AVOCADOS HASS OG EACH	97752.460
11276	9	PECANS HALVES ORGANIC	75230.922
14054	9	SPRING WATER GLASS	48761.180
8529	9	LEMONS OG	42596.400

```
fig = px.bar(mon_sales_Earth_2021_9,
             x = 'Reciept Alias',
             y = "Sales",
             )
#fig.update_yaxes(range=[0, 5000])

fig.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```



When we narrow sales generated form September, most of the sales come from the Healthy Dege Water Alkaline 9+, which dominates the market basket portions with $5M sales, significantly. This pattern could marked some significant marketing campaigns held upon the Healthy Edge Water Alkaline 9+ during that month.

Sales by loyalty tiers

```
loyal_sales_Earth_2021 = basket_data_Earth[(basket_data_Earth['year'] == 2021) & (basket_data_Earth['Loyalty Customer?'] == 1)].
loyal_sales_Earth_2022 = basket_data_Earth[(basket_data_Earth['year'] == 2022) & (basket_data_Earth['Loyalty Customer?'] == 1)].
loyal_sales_Earth_2023 = basket_data_Earth[(basket_data_Earth['year'] == 2023) & (basket_data_Earth['Loyalty Customer?'] == 1)].
```

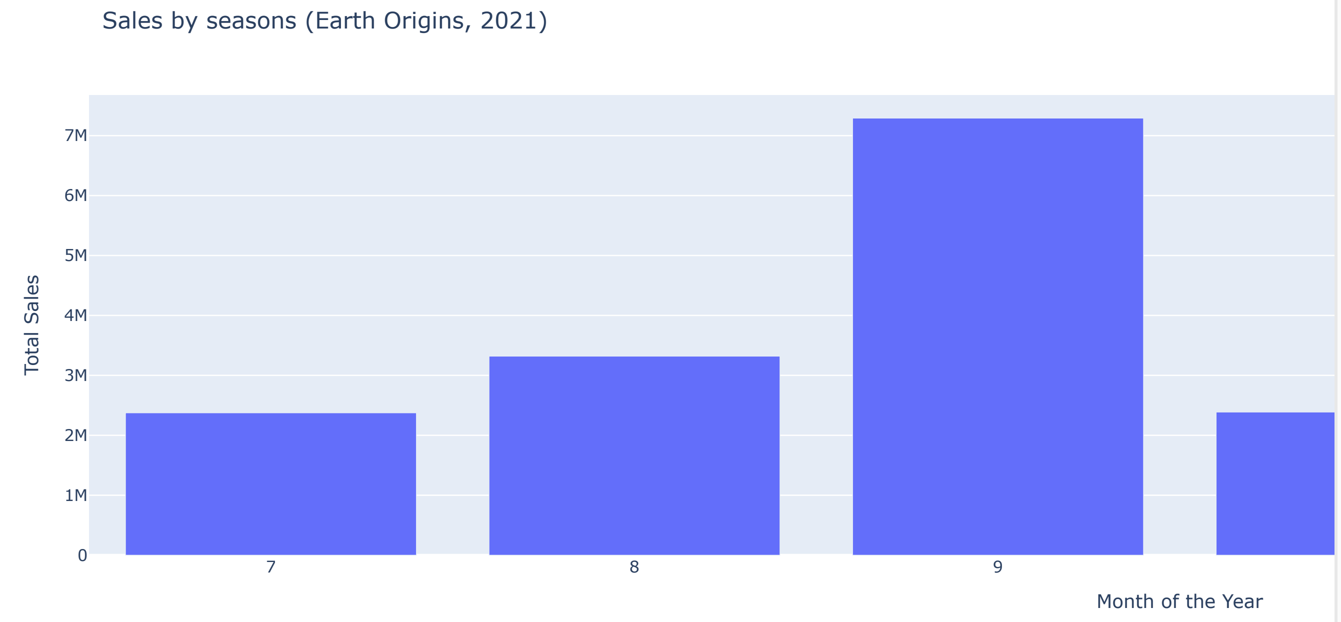
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

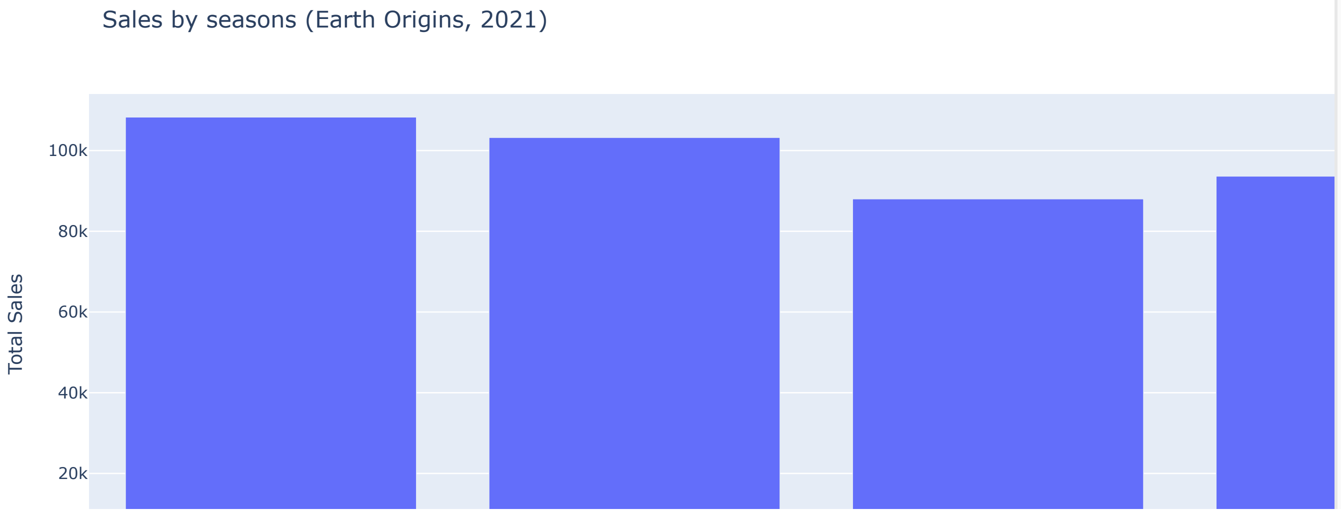
```
guest_sales_Earth_2021 = basket_data_Earth[(basket_data_Earth['year'] == 2021) & (basket_data_Earth['Loyalty Customer?'] == 0)].
guest_sales_Earth_2022 = basket_data_Earth[(basket_data_Earth['year'] == 2022) & (basket_data_Earth['Loyalty Customer?'] == 0)].
guest_sales_Earth_2023 = basket_data_Earth[(basket_data_Earth['year'] == 2023) & (basket_data_Earth['Loyalty Customer?'] == 0)].
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
fig = px.bar(loyal_sales_Earth_2021,
             x='month',
             y= 'Sales',
             title='Sales by seasons (Earth Origins, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig_2 = px.bar(guest_sales_Earth_2021,
             x='month',
             y= 'Sales',
             title='Sales by seasons (Earth Origins, 2021)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig.show()
fig_2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```

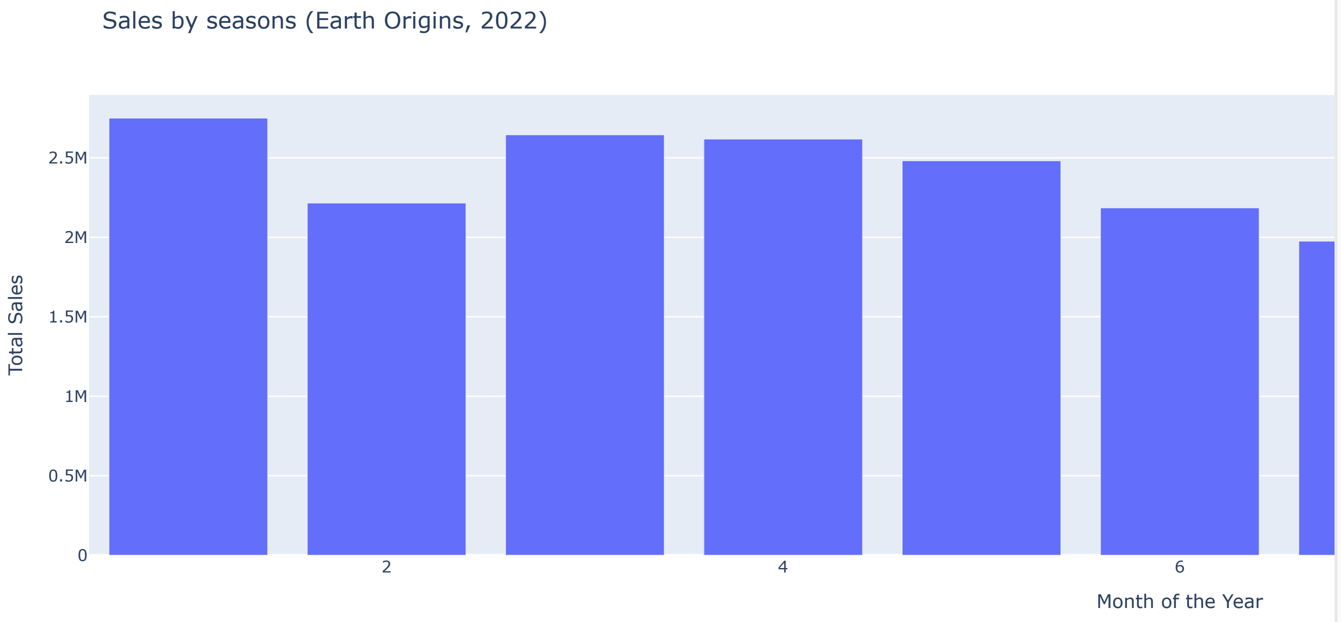


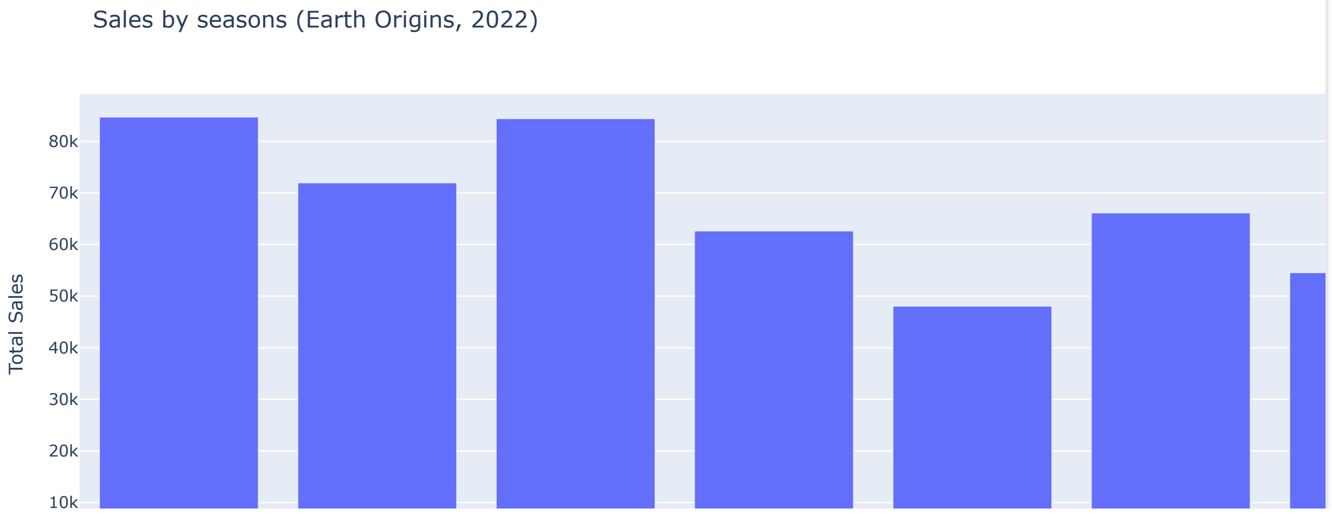


The graph implied to the fact that the loyalty member got higher motivation to purchase water than the guest, which could means that there were some exclusive member sales campaign held in September 2021 for store members.

```
fig = px.bar(loyal_sales_Earth_2022,
             x='month',
             y= 'Sales',
             title='Sales by seasons (Earth Origins, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig_2 = px.bar(guest_sales_Earth_2022,
             x='month',
             y= 'Sales',
             title='Sales by seasons (Earth Origins, 2022)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig.show()
fig_2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```

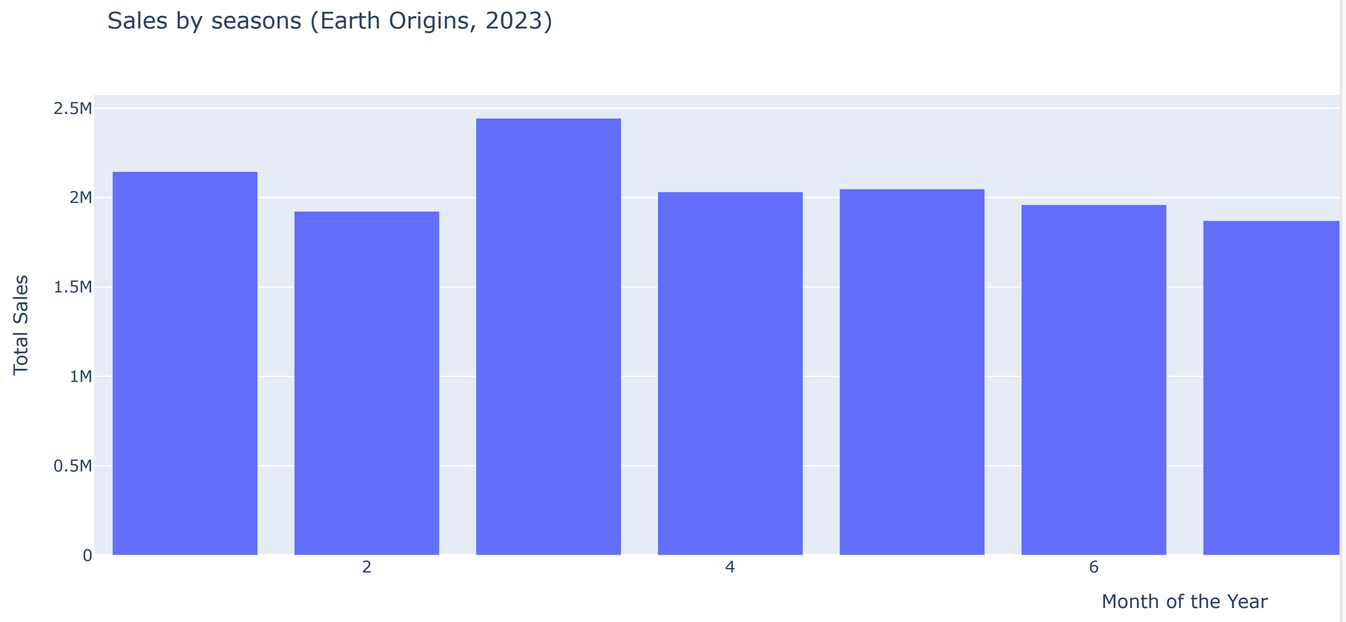


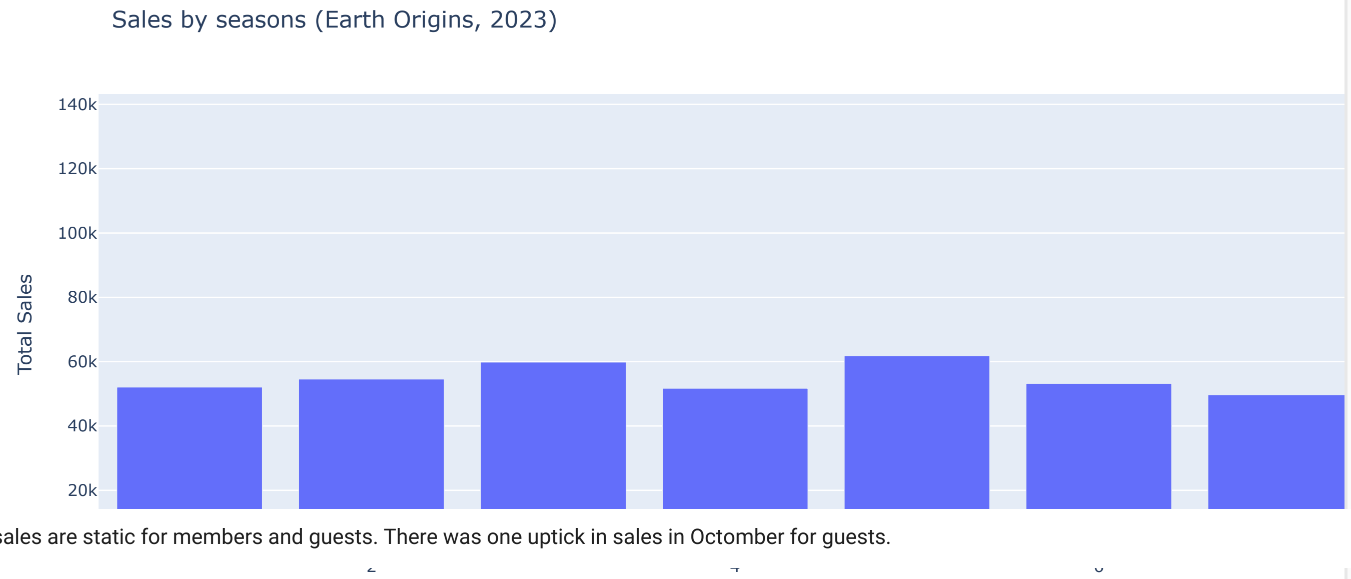


Sales for member and guest both follows strong seasonalities in the first 2 quarters and slightly decreased in the next 2 quarters.

```
fig = px.bar(loyal_sales_Earth_2023,
             x='month',
             y= 'Sales',
             title='Sales by seasons (Earth Origins, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig_2 = px.bar(guest_sales_Earth_2023,
             x='month',
             y= 'Sales',
             title='Sales by seasons (Earth Origins, 2023)',
             labels={'Sales': 'Total Sales', 'month': 'Month of the Year'},
             hover_data=['month', 'Sales'])

fig.show()
fig_2.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```





Both sales are static for members and guests. There was one uptick in sales in Octomber for guests.

Top sold item for each tier

```
item_top_5 = basket_data_Earth.groupby(['Reciept Alias', 'Loyalty Customer?'])['Sales'].sum().reset_index()
loyal_item_top_5= item_top_5[item_top_5['Loyalty Customer?'] == 1].sort_values(by='Sales', ascending=False).head(5)
guest_item_top_5 = item_top_5[item_top_5['Loyalty Customer?'] == 0].sort_values(by='Sales', ascending=False).head(5)
print(loyal_item_top_5)
print(guest_item_top_5)
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`

                        Reciept Alias  Loyalty Customer?        Sales
20564  HEALTHY EDGE WATER ALKALINE 9+                  1  6225406.860
2488            AVOCADOS HASS OG EACH                  1  2725852.250
40529               SPRING WATER GLASS                  1  1523816.490
24481                        LEMONS OG                  1  1121303.923
7669                        CARROTS OG                  1   905747.370
               Reciept Alias  Loyalty Customer?     Sales
23202    KENT/KEITT MANGO OG                  0  97712.50
2487   AVOCADOS HASS OG EACH                  0  54028.47
```

```
40528     SPRING WATER GLASS                0  52117.41
24480              LEMONS OG                0  27975.71
40201          SPECIAL ORDER                0  20702.24
```

EDA for Akins

```
sales_Akins_2021 = basket_data_Akins[basket_data_Akins['year'] == 2021].groupby(['Category','Subcategory']).agg({'Sales': "sum"}
sales_Akins_2022 = basket_data_Akins[basket_data_Akins['year'] == 2022].groupby(['Category','Subcategory']).agg({'Sales': "sum"}
sales_Akins_2023 = basket_data_Akins[basket_data_Akins['year'] == 2023].groupby(['Category','Subcategory']).agg({'Sales': "sum"}
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
#check the table structure
sales_Akins_2023
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

		Sales
Category	**Subcategory**	
ANTIOXIDANTS & ANTIAGING	**AMINO ACIDS**	1870.29
	COQ10 & UBIQUINOL	72564.06
	DHEA & PREGNENOLONE	10451.61
	HORMONES & ANTI-AGING	7143.76
	MINERALS	24.99
...	**...**	...
YOGURT & KEFIR	**DAIRY YOGURT CUPS**	3432.96
	GREEK & SPECIALTY YOGURT	4063.43
	NON-DAIRY YOGURT & KEFIR	22281.47
	YOGURT & KEFIR	15848.89
	YOGURT & KEFIR DRINKS	14229.45

777 rows × 1 columns

```
sales_Akins_2021.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 646 entries, ('ANTIOXIDANTS & ANTIAGING', 'AMINO ACIDS') to ('YOGURT & KEFIR', 'YOGURT & KEFIR DRINKS')
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Sales   646 non-null    float64
dtypes: float64(1)
memory usage: 11.4+ KB
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

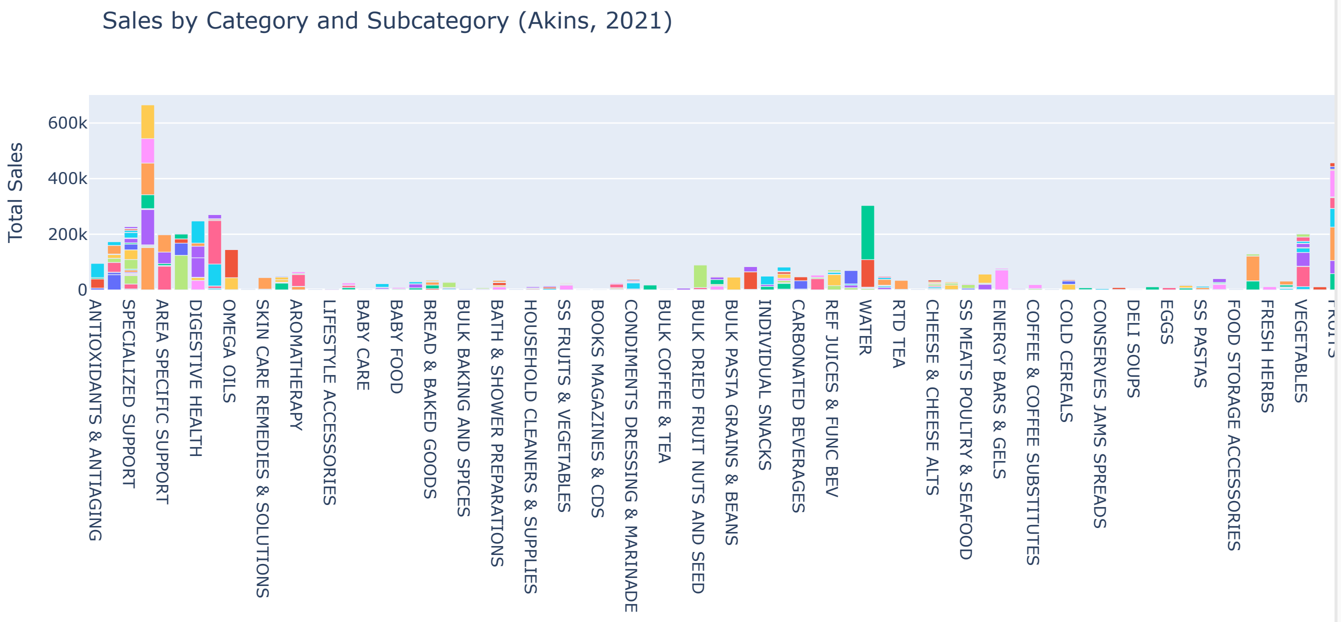
identify Sales from each product categories

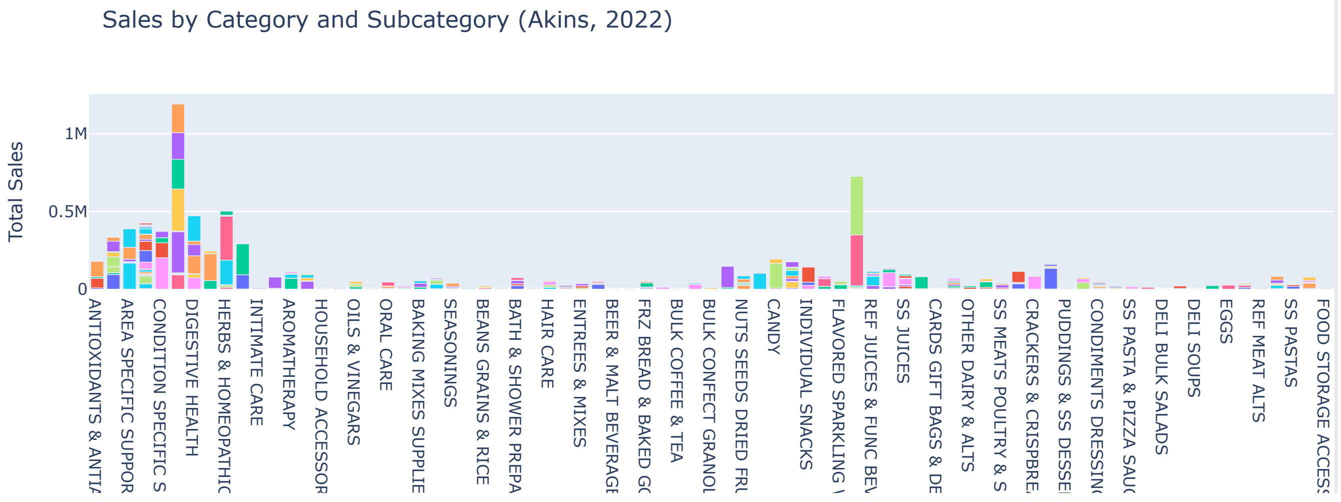
```
#reset the index of each subtable
sales_Akins_2021 = sales_Akins_2021.reset_index()
sales_Akins_2022 = sales_Akins_2022.reset_index()
sales_Akins_2023 = sales_Akins_2023.reset_index()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
# Create the bar chart using Plotly Express

fig1 = px.bar(sales_Akins_2021,
             x='Category',
             y='Sales',
             color='Subcategory',  # Color bars by subcategory
             title='Sales by Category and Subcategory (Akins, 2021)',
             labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
             hover_data=['Subcategory', 'Sales'])

fig2 = px.bar(sales_Akins_2022,
             x='Category',
             y='Sales',
             color='Subcategory',  # Color bars by subcategory
             title='Sales by Category and Subcategory (Akins, 2022)',
             labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
             hover_data=['Subcategory', 'Sales'])

fig3 = px.bar(sales_Akins_2023,
             x='Category',
             y='Sales',
             color='Subcategory',  # Color bars by subcategory
             title='Sales by Category and Subcategory (Akins, 2023)',
             labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
             hover_data=['Subcategory', 'Sales'])

fig1.show()
fig2.show()
fig3.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```





Vitamins dominate the top selling catagories throughout the 3 years, and distirbuted evenly among multiple product types.

identify seasonality patterns in sales

```
mon_sales_Akins_2021 = basket_data_Akins[basket_data_Akins['year'] == 2021].groupby(['month']).agg({'Sales': "sum"}).reset_index
mon_sales_Akins_2022 = basket_data_Akins[basket_data_Akins['year'] == 2022].groupby(['month']).agg({'Sales': "sum"}).reset_index
mon_sales_Akins_2023 = basket_data_Akins[basket_data_Akins['year'] == 2023].groupby(['month']).agg({'Sales':"sum"}).reset_index(
```

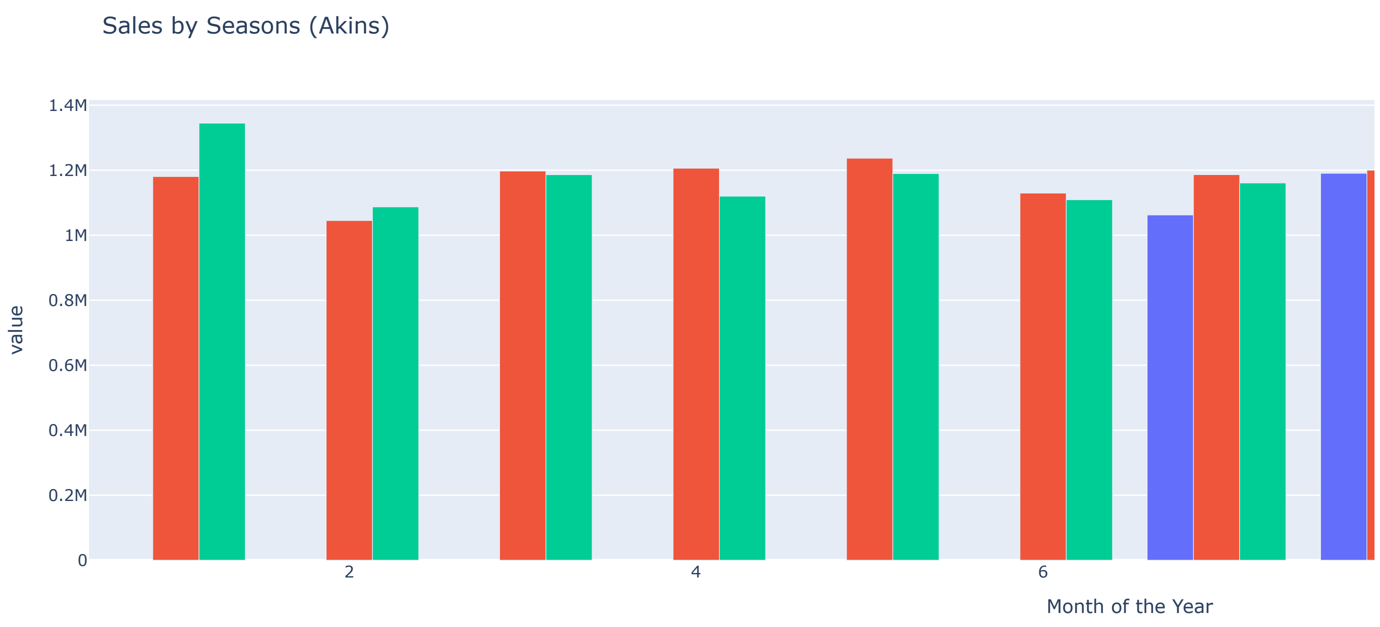
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
mon_sales_Akins = mon_sales_Akins_2021.merge(mon_sales_Akins_2022, on='month',how='outer',left_index=False,right_index=False).re
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
fig = px.bar(mon_sales_Akins,
             x='month',
             y =['2021 Sales','2022 Sales','2023 Sales'],
             title = 'Sales by Seasons (Akins)',
             barmode='group',
             labels = {'Sales': 'Total Sales', 'month': 'Month of the Year'}
             )
fig.show()
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```



Janurary tends to be the month that generates the most sales. On yearly basis, the sales growth tend to remain static.

sales diff by loyalty tiers

```
loyal_sales_Akins_2021 = basket_data_Akins[(basket_data_Akins['year'] == 2021) & (basket_data_Akins['Loyalty Customer?'] == 1)].
loyal_sales_Akins_2022 = basket_data_Akins[(basket_data_Akins['year'] == 2022) & (basket_data_Akins['Loyalty Customer?'] == 1)].
loyal_sales_Akins_2023 = basket_data_Akins[(basket_data_Akins['year'] == 2023) & (basket_data_Akins['Loyalty Customer?'] == 1)].
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
guest_sales_Akins_2021 = basket_data_Akins[(basket_data_Akins['year'] == 2021) & (basket_data_Akins['Loyalty Customer?'] == 0)].
guest_sales_Akins_2022 = basket_data_Akins[(basket_data_Akins['year'] == 2022) & (basket_data_Akins['Loyalty Customer?'] == 0)].
guest_sales_Akins_2023 = basket_data_Akins[(basket_data_Akins['year'] == 2023) & (basket_data_Akins['Loyalty Customer?'] == 0)].
```

```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipykernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
fig = px.bar(loyal_sales_Akins_2021,
             x='month',
             y= 'Sales',
```