```
#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
```

```
/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`
```

```
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 `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
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 `transform_cell` automatically in the future. Please pass the result to `transformed_cell`

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=Tr
```

```
# 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 `transform_cell` automatically in the future. Please pass the result to `transformed_cell`

<ipython-input-110-c7f3872205e6>:4: DtypeWarning:

Columns (11) have mixed types. Specify dtype option on import or set low_memory=False.
```

```
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 `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
basket_data_raw.head()
```

```
/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`
```

	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 `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
#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 `transform_cell` automatically in the future. Please pass the result to `transformed_cell`

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 `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

```
basket_data_Akins.head(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`
```

	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 `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

	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	20
5	9.99	1/11/22	1.19	9.99	11.89	REFRIGERATED	1	11	20
6	4.29	1/11/22	1.00	4.29	4.29	PRODUCE	1	11	20
7	17.99	1/11/22	1.00	17.99	17.99	SUPPLEMENTS	1	11	20
8	2.39	1/11/22	1.00	2.39	2.39	PRODUCE	1	11	20

```
basket_data_Chamberlins.head(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`
```

	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

AI

```
/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`
```

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']).agg
sales_Chamberlins_2022 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2022].groupby(['Category','Subcategory']).agg
sales_Chamberlins_2023 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2023].groupby(['Category','Subcategory']).agg
```

```
/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`
```

```
#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 `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

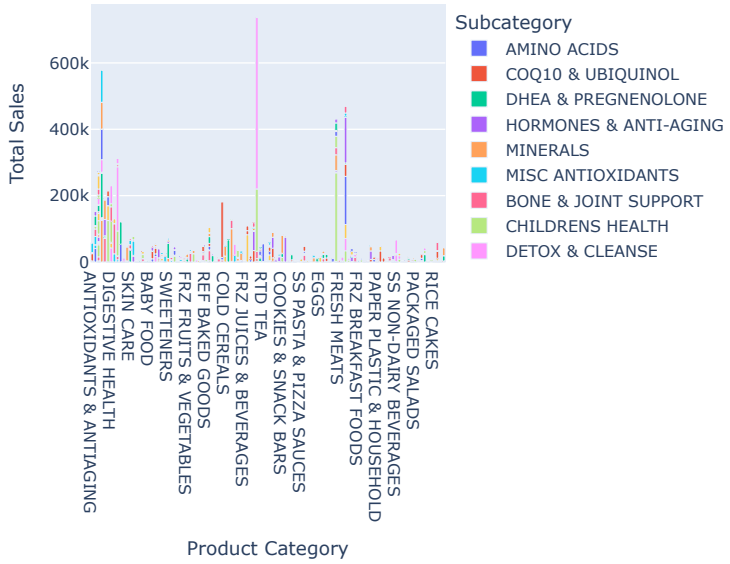
Category Level Performance

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

fig1 = px.bar(sales_Chamberlins_2021,
```

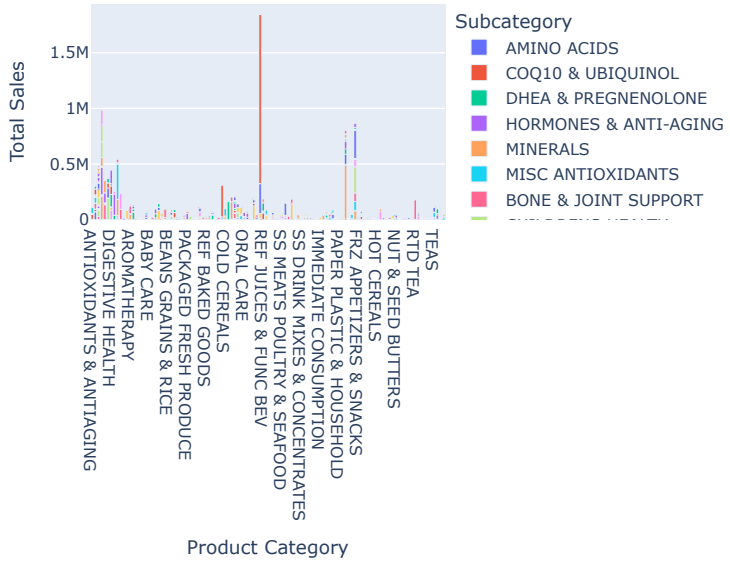
```
                x='Category',
                y='Sales',
                color='Subcategory',  # Color bars by subcategory
                title='Sales by Category and Subcategory (Chamberlins, 2021)',
                labels={'Sales': 'Total Sales', 'Category': 'Product Category'},
                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 `transform_cell` automatically in the future. Please pass the result to `transformed_ce
```

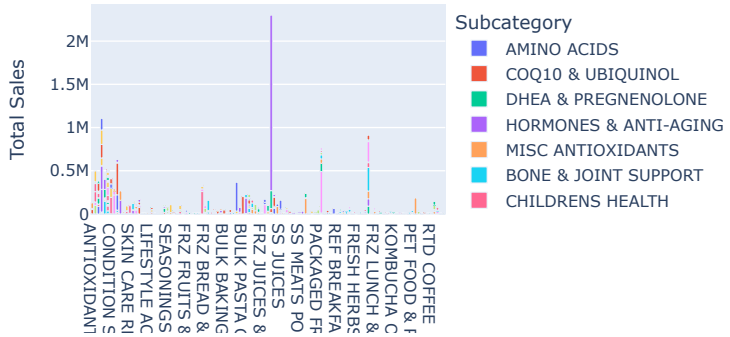
Sales by Category and Subcategory (Chamberlins, 2021)



Sales by Category and Subcategory (Chamberlins, 2022)



Sales by Category and Subcategory (Chamberlins, 2023)



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 line of Category WATER generating the most revenues. Therefore, we can clearly conclude that on the macro level, the largest share of profitability lies in the Packaged Water Subcategory. However, one 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 further valdiate the cause of profitability, we need to break down metrics by adding more dimensional variables.

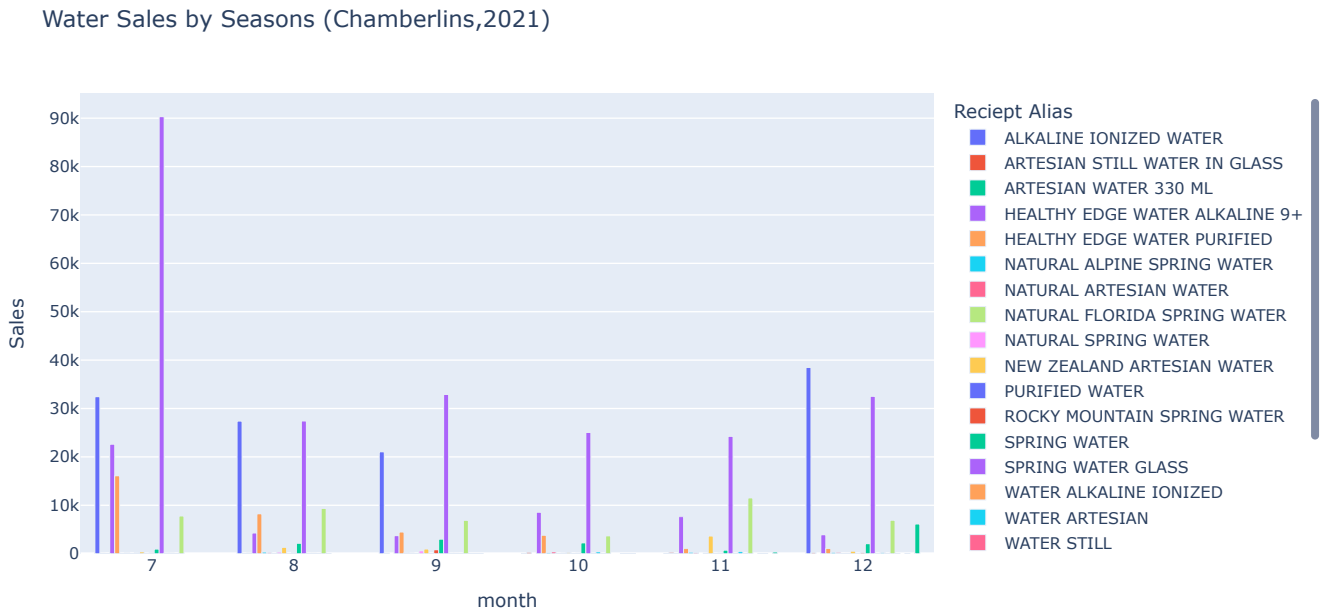
Monthly Sales Trends of Water Category

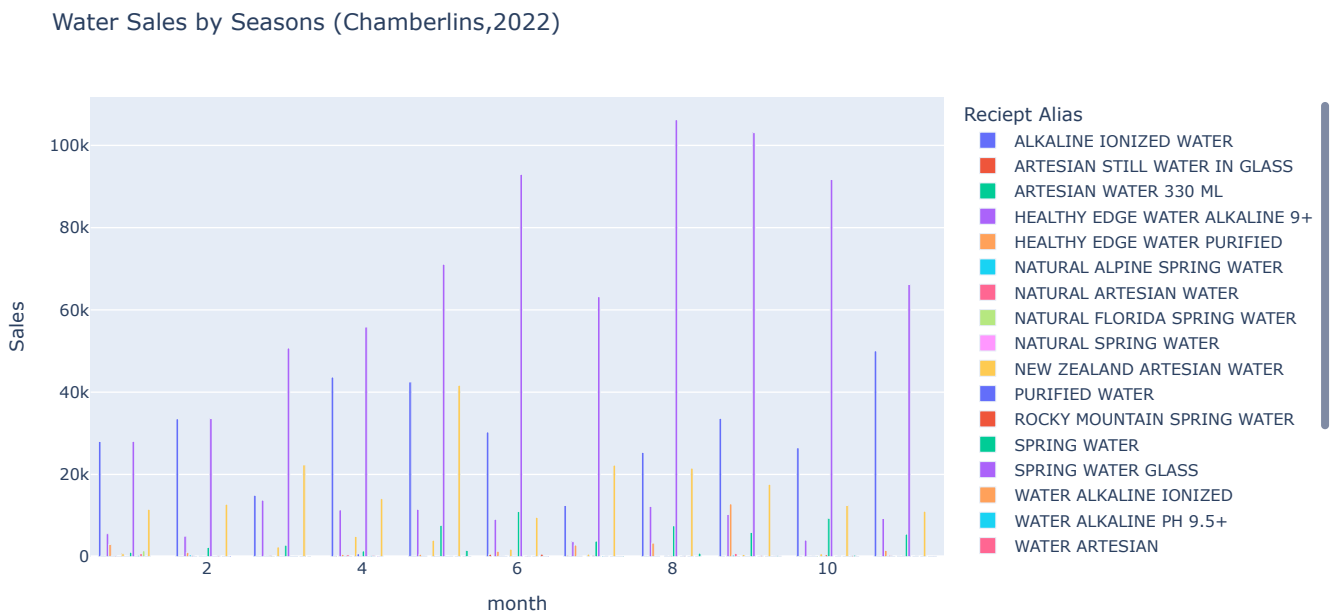
```
mon_sales_Chamberlins_water_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2021) & (basket_data_Chamberlins[
mon_sales_Chamberlins_water_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins[
mon_sales_Chamberlins_water_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['year']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

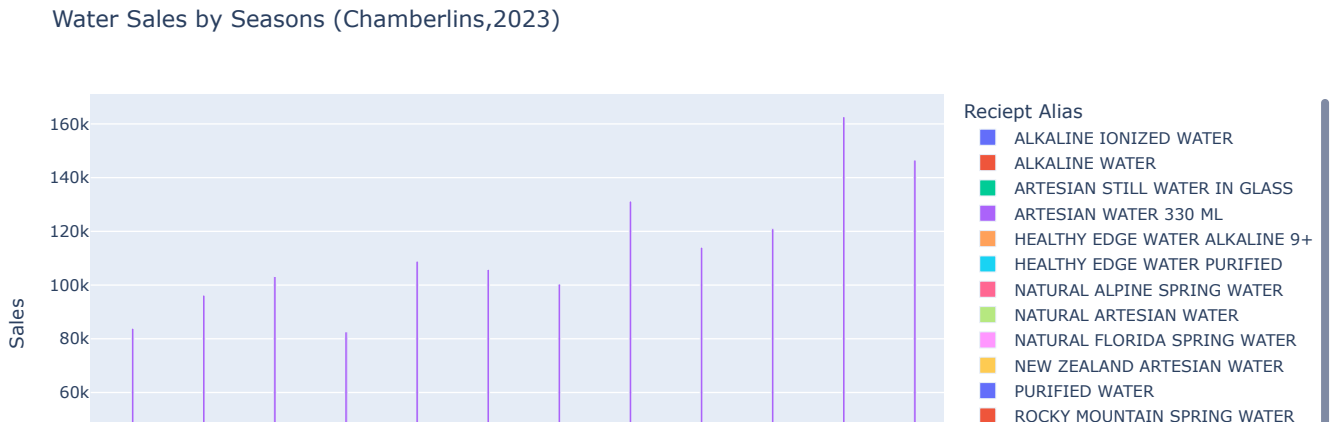
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

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

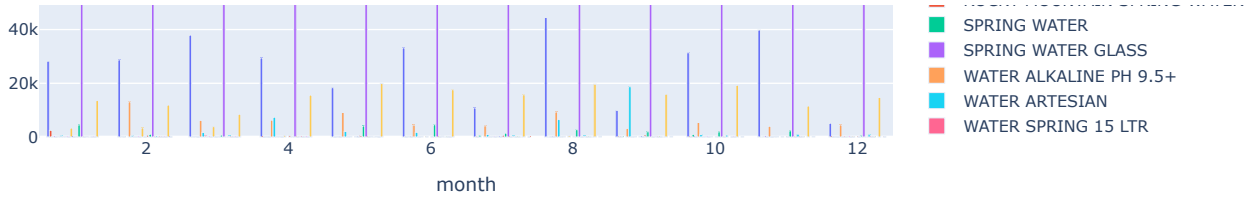
```
fig1 = px.bar(mon_sales_Chamberlins_water_2021,
             x = 'month',
             y ='Sales',
             title ='Water Sales by Seasons (Chamberlins,2021)',
             color = 'Reciept Alias',
             barmode='group'
            )

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 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, marketing campaigns should be ideally assigned towards those second-place substitudes as they hold among the highest growth potentials

Recommendation:

By sugesstion, regional a/b testing of marketing promotional campaigns should be implemented on second most sold merchandise to evaluate the impact.

Sales Trend at seasonal level

---

1. Annual Revenue

```
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`
```

```
mon_sales_Chamberlins
```

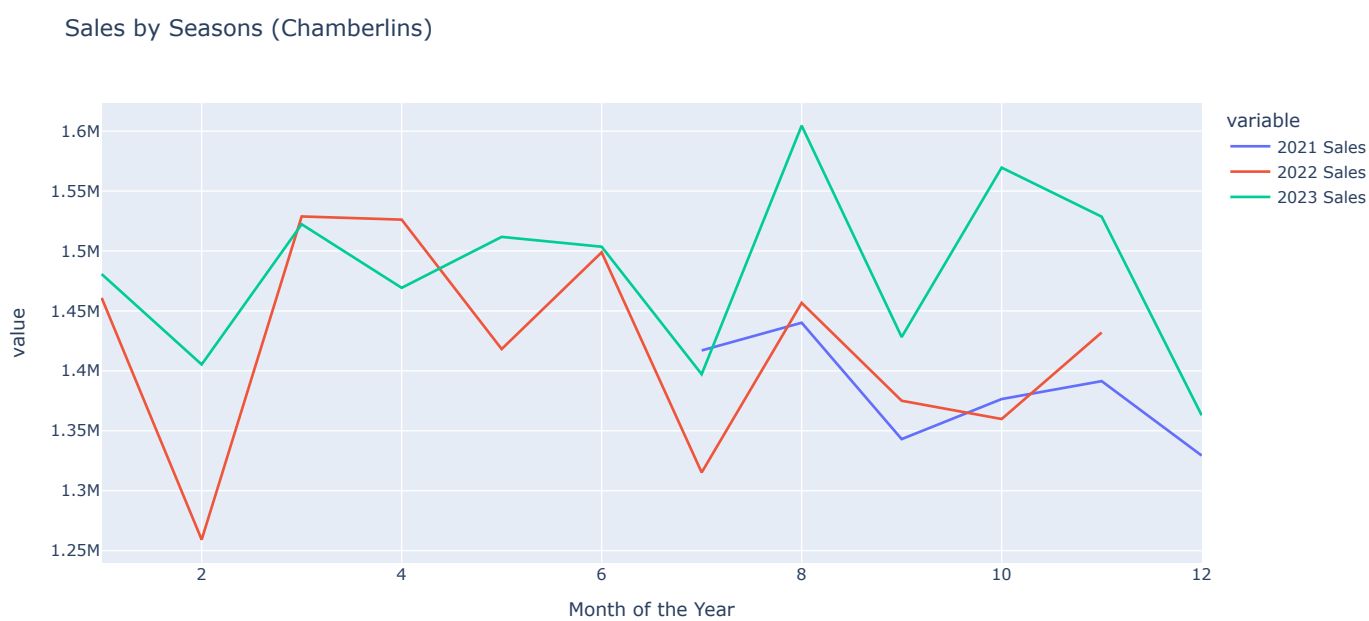
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

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

	month	2021 Sales	2022 Sales	2023 Sales
0	1	NaN	1.460885e+06	1.480727e+06
1	2	NaN	1.259144e+06	1.405289e+06
2	3	NaN	1.528827e+06	1.522302e+06
3	4	NaN	1.526177e+06	1.469270e+06
4	5	NaN	1.418176e+06	1.511767e+06
5	6	NaN	1.498910e+06	1.503573e+06
6	7	1.417011e+06	1.315208e+06	1.397186e+06
7	8	1.440144e+06	1.456728e+06	1.604573e+06
8	9	1.343052e+06	1.375056e+06	1.428210e+06
9	10	1.376470e+06	1.359800e+06	1.569559e+06
10	11	1.391417e+06	1.432009e+06	1.528633e+06
11	12	1.329322e+06	NaN	1.362907e+06

mon_sales_Chamberlins New interactive sheet

```
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/ipkernel.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 identify a seasonality pattern that March and August tend to have the most sales generated through the year. The sales stagnated from year 2021 to year 2022 but ramped rapidly in later year 2023.

This phenomenon can be attributed to the change of consuming behaviors post-pandemic. As consumer's financial status grow static, the intuition of grocery shopping increased.

2. Annual Order Volume

```
mon_volume_Chamberlins_2021 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2021].groupby(['month'])['Receipt Numbe
mon_volume_Chamberlins_2022 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2022].groupby(['month'])['Receipt Numbe
mon_volume_Chamberlins_2023 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2023].groupby(['month'])['Receipt Numbe
```

```
/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_volume_Chamberlins = mon_volume_Chamberlins_2021.merge(mon_volume_Chamberlins_2022, on='month',how='outer',left_index=False,
```

```
/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_volume_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`
```

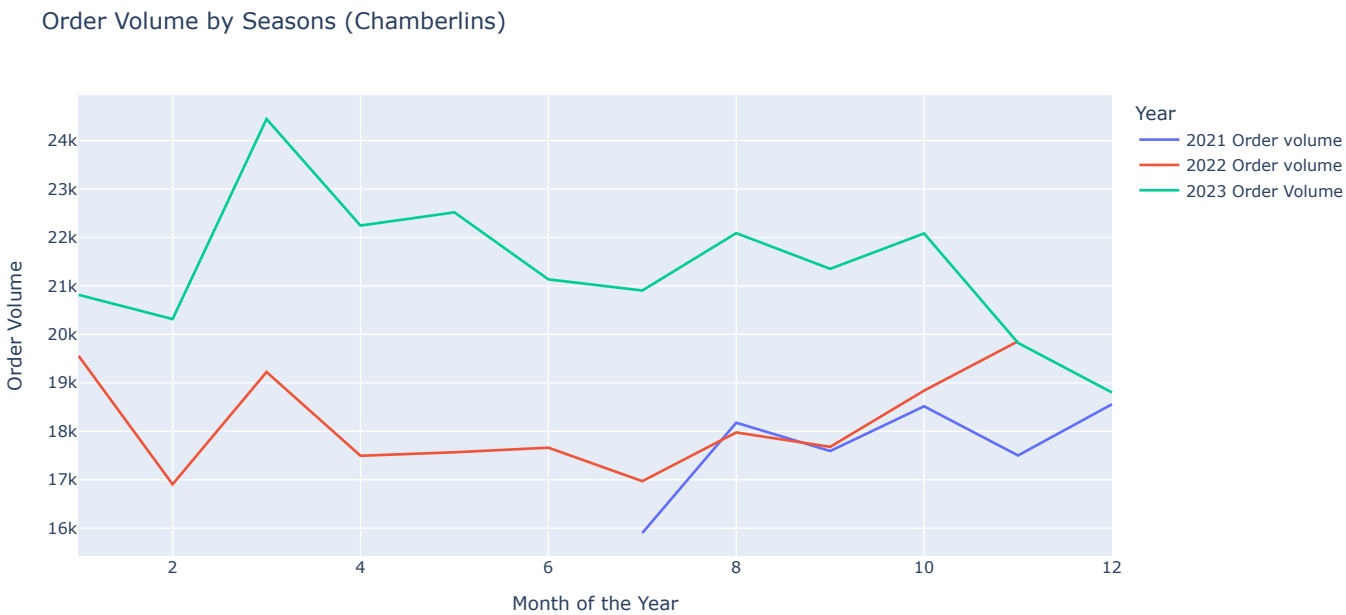
	month	2021 Order volume	2022 Order volume	2023 Order Volume
0	1	NaN	19556.0	20817
1	2	NaN	16906.0	20316
2	3	NaN	19224.0	24446
3	4	NaN	17495.0	22247
4	5	NaN	17567.0	22519
5	6	NaN	17662.0	21136
6	7	15901.0	16971.0	20906
7	8	18177.0	17976.0	22090
8	9	17593.0	17679.0	21353
9	10	18516.0	18840.0	22084
10	11	17501.0	19859.0	19824
11	12	18559.0	NaN	18801

mon_volume_Chamberlins New interactive sheet

```
mon_volume_Chamberlins_long = pd.melt(
    mon_volume_Chamberlins,
    id_vars=['month'],  # Keep 'month' as the identifier
    value_vars=['2021 Order volume', '2022 Order volume', '2023 Order Volume'],  # Columns to melt
    var_name='Year',  # Name for the new column containing year
    value_name='Order Volume'  # Name for the new column containing order volume
)

# Create the Plotly Express line chart
fig = px.line(
    mon_volume_Chamberlins_long,
    x='month',
    y='Order Volume',
    color='Year',  # Color lines by year
    title='Order Volume by Seasons (Chamberlins)',
    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`
```



We can clearly identify that year 2023 has the most order volume sold, indicating strong YoY growth potential . In observance of the sales volume uptick in March for both year 2022 and 2023, it can be assumed that the demand peaks around the 1st quarter of the year and slowly decrease until the 3rd quarter of that year.

3. Average Order Value

```
mon_aov_Chamberlins_2021 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2021].groupby(['month','Receipt Number']).
mon_aov_Chamberlins_2022 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2022].groupby(['month','Receipt Number']).
mon_aov_Chamberlins_2023 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2023].groupby(['month','Receipt Number']).
```

```
/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_aov_Chamberlins_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	Sales
0	7	89.114608
1	8	79.228933
2	9	76.340125
3	10	74.339473
4	11	79.504977
5	12	71.626835

mon_aov_Chamberlins_2021 New interactive sheet

```
mon_aov_Chamberlins = mon_aov_Chamberlins_2021.merge(mon_aov_Chamberlins_2022, on='month',how='outer',left_index=False,right_inc
```

```
/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_aov_Chamberlins
```

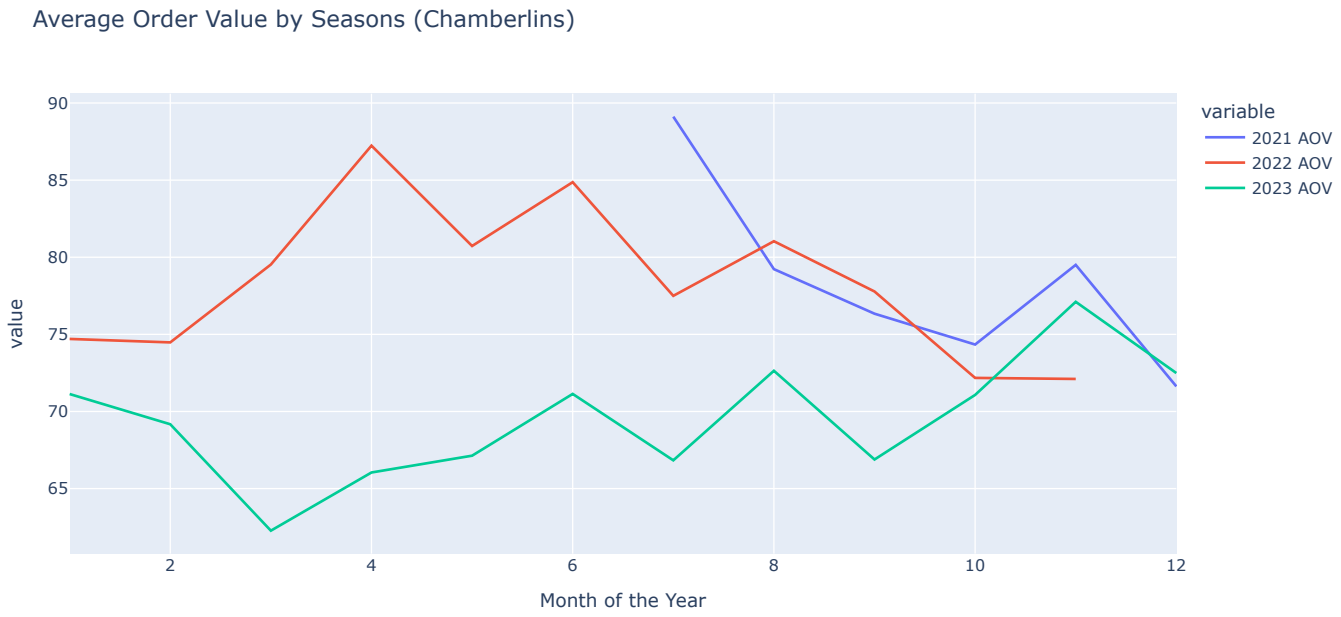
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell`
```

	month	2021 AOV	2022 AOV	2023 AOV
0	1	NaN	74.702633	71.130667
1	2	NaN	74.479099	69.171537
2	3	NaN	79.526983	62.272031
3	4	NaN	87.235032	66.043496
4	5	NaN	80.729558	67.132961
5	6	NaN	84.866370	71.138025
6	7	89.114608	77.497357	66.831838
7	8	79.228933	81.037383	72.637972
8	9	76.340125	77.779074	66.885697
9	10	74.339473	72.176220	71.072208
10	11	79.504977	72.108796	77.110234
11	12	71.626835	NaN	72.491197

mon_aov_Chamberlins New interactive sheet

```
# Create the Plotly Express line chart
fig = px.line(
      mon_aov_Chamberlins,
      x='month',
      y=['2021 AOV',    '2022 AOV', '2023 AOV'],

      title='Average Order Value by Seasons (Chamberlins)',
      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`
```



We can observed that the AOV stays in the range of $60-90 over the three years, with the year 2021 and 2022 topped the average value and the year 2023 slightly slacked behind. This could makred a slight price change from pandemic eyars to the post-pandemic year.

Sales by loyalty Status

```
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/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_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/ipkernel.py:283: DeprecationWarning:

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

```
Tier_sales_Chamberlins_2021 = loyal_sales_Chamberlins_2021.merge(guest_sales_Chamberlins_2021, on='month',how='outer',left_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`
```

```
Tier_sales_Chamberlins_2021
```

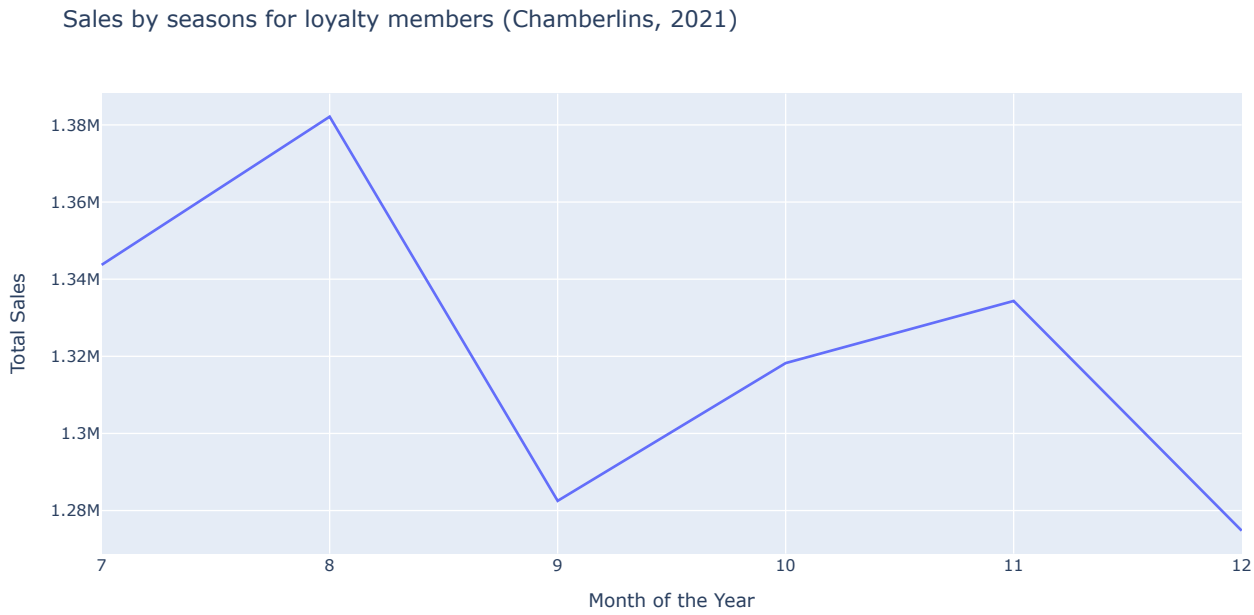
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

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

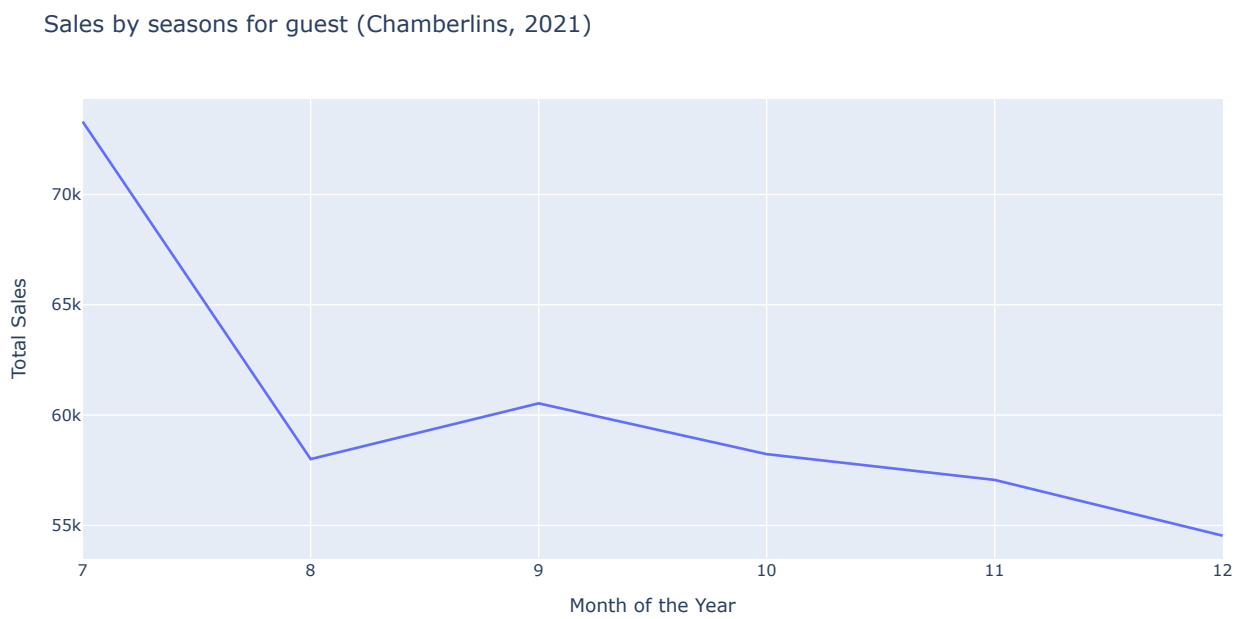
	month	Loyal Sales	Guest Sales
0	7	1.343710e+06	73301.2789
1	8	1.382139e+06	58005.5532
2	9	1.282519e+06	60533.0875
3	10	1.318239e+06	58230.5557
4	11	1.334355e+06	57061.9477
5	12	1.274787e+06	54535.0212

Tier_sales_Chamberlins_2021 New interactive sheet

```
fig = px.line(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.line(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/ipkernel.py:283: DeprecationWarning:

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



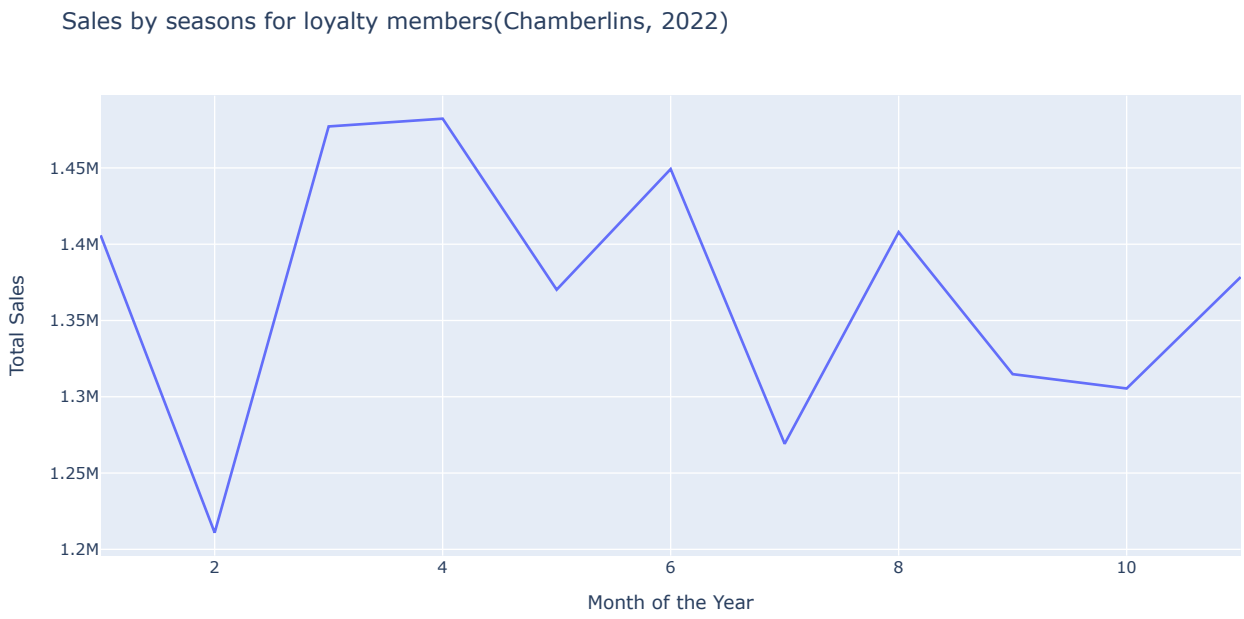


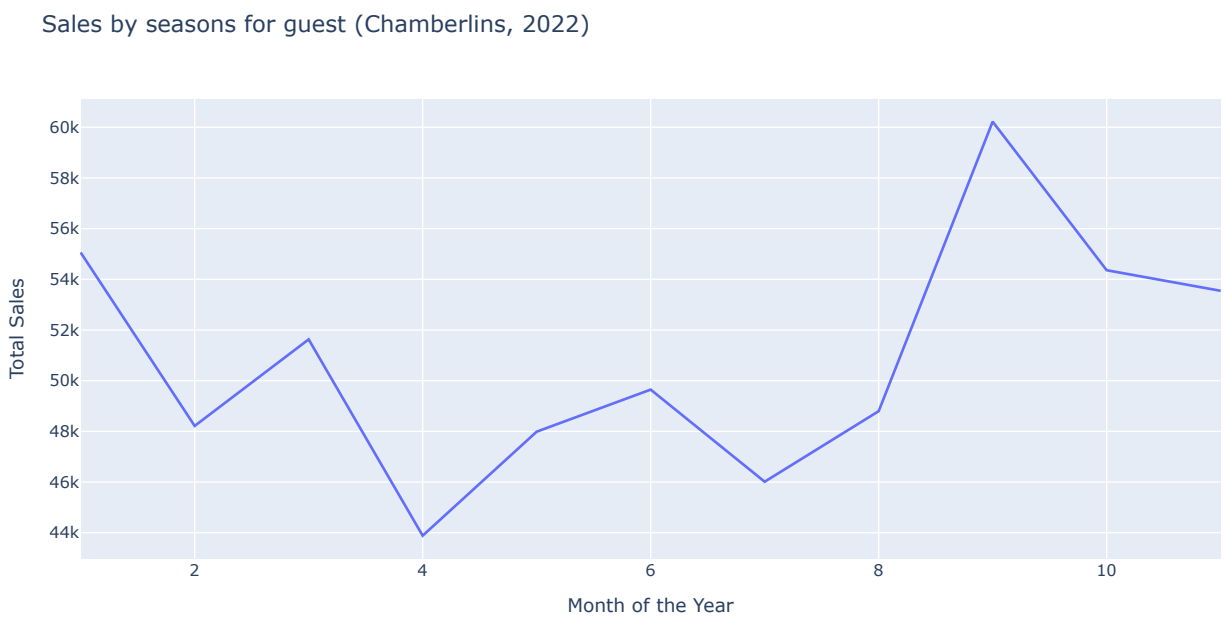
In year 2021, sales were trended with seasonal patterns for the loyalty members, whereas the sales from guests decreased month to month.

```
fig = px.line(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.line(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
```



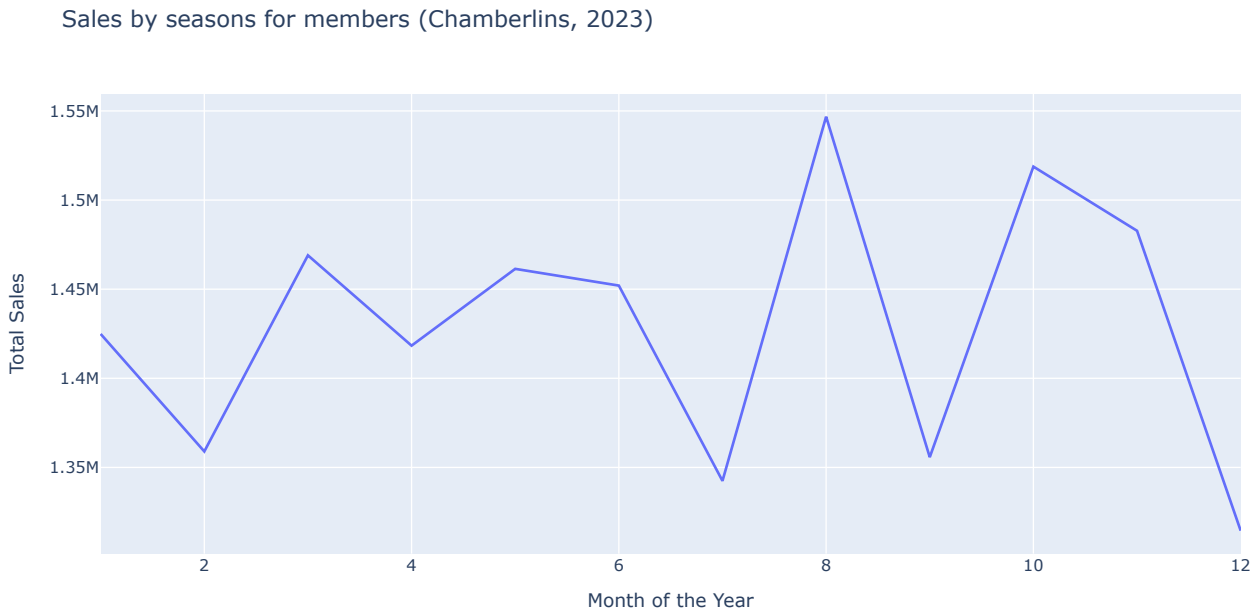


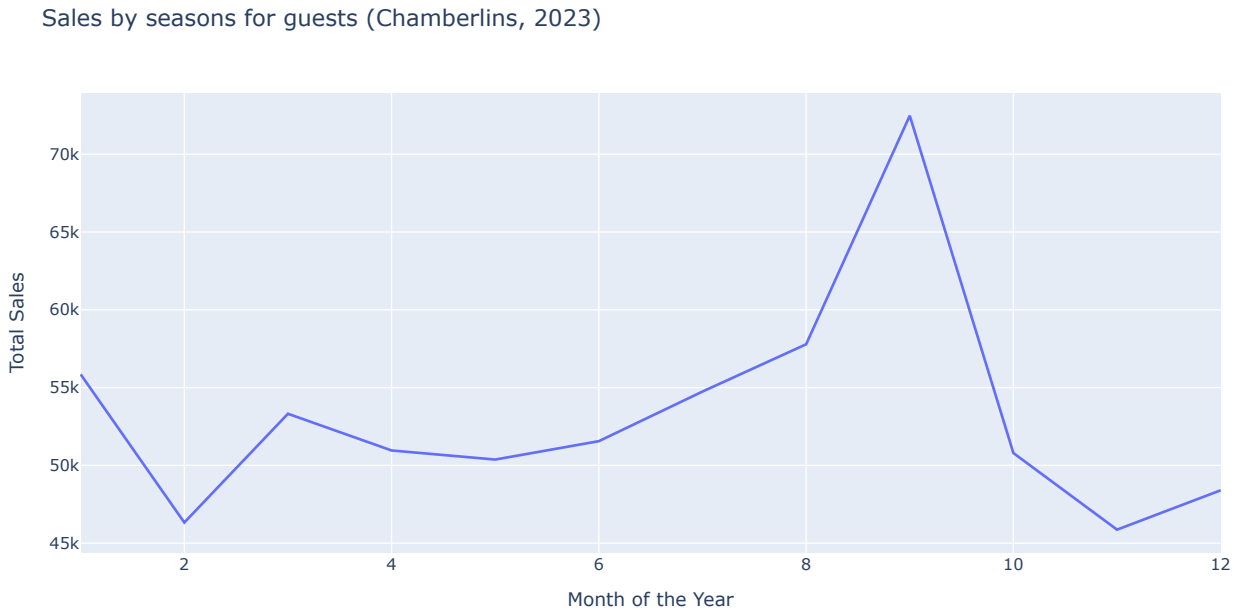
In year 2022, the sales form loyalty memebers peaked in mArch and April, and remained static over the year, whereas the sales from guest had a slight uptick in September and declined in Octomber.

```
fig = px.line(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.line(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/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 2023, sales indicated strong seasonalities from members. September sales has increased shortly for guests but edeclined shortly after to the lowest point in November..

Average Order Value by Loyalty Status

```
mon_aov_Chamberlins_member_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2021) & (basket_data_Chamberlins['
mon_aov_Chamberlins_member_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins['
mon_aov_Chamberlins_member_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_aov_Chamberlins_guest_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2021) & (basket_data_Chamberlins['L
mon_aov_Chamberlins_guest_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins['L
mon_aov_Chamberlins_guest_2023 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2023) & (basket_data_Chamberlins['L
```

```
/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_aov_Chamberlins_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	Sales
0	7	89.114608
1	8	79.228933
2	9	76.340125
3	10	74.339473
4	11	79.504977
5	12	71.626835

mon_aov_Chamberlins_2021 New interactive sheet

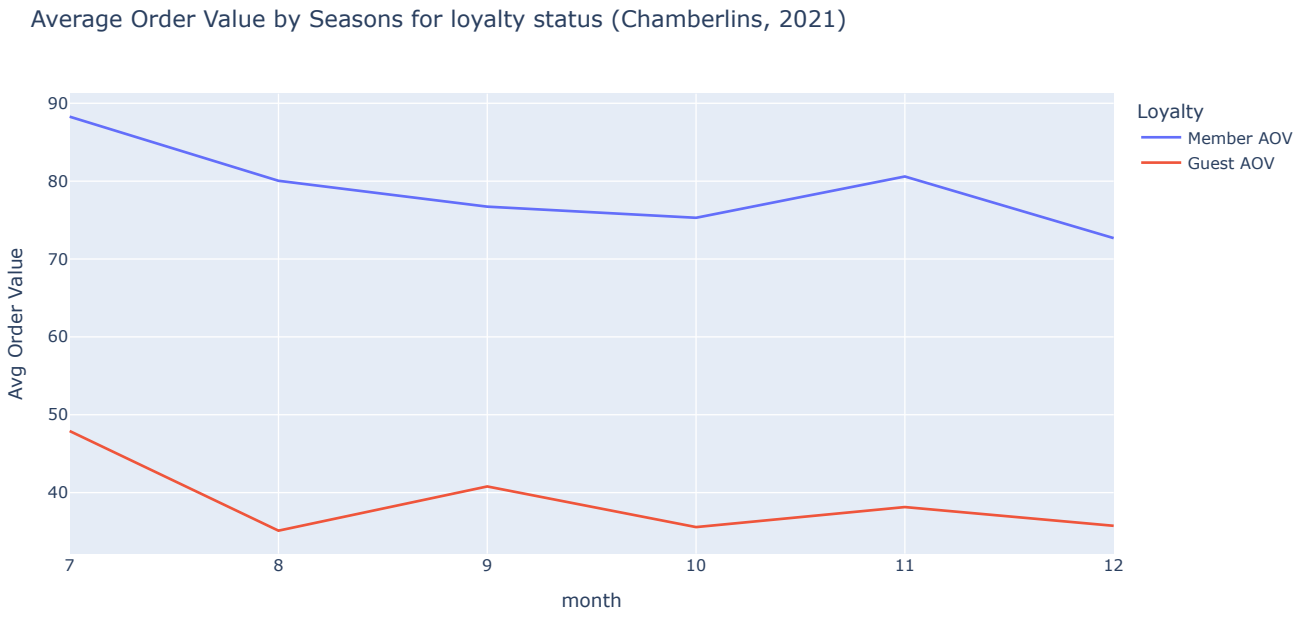
```
mon_aov_Chamberlins_2021 = mon_aov_Chamberlins_member_2021.merge(mon_aov_Chamberlins_guest_2021, on='month',how='outer',left_inc
mon_aov_Chamberlins_2022 = mon_aov_Chamberlins_member_2022.merge(mon_aov_Chamberlins_guest_2021, on='month',how='outer',left_inc
mon_aov_Chamberlins_2023 = mon_aov_Chamberlins_member_2021.merge(mon_aov_Chamberlins_guest_2023, on='month',how='outer',left_inc
```

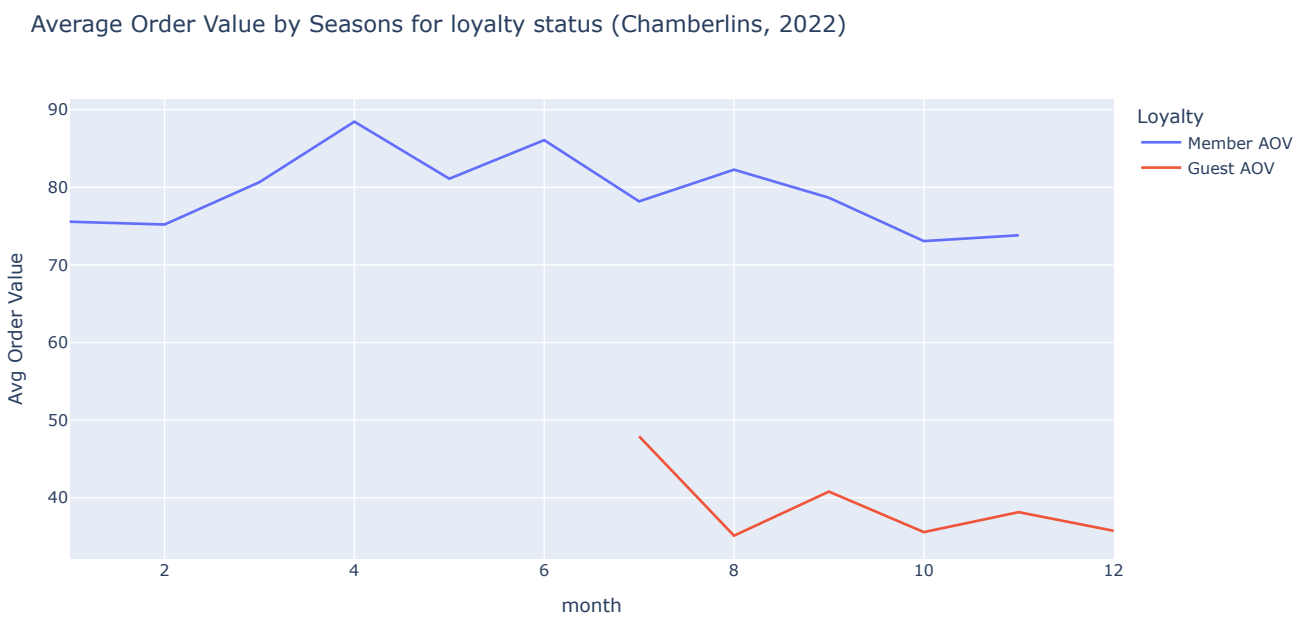
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

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

```
mon_aov_Chamberlins_2021_long = pd.melt(
    mon_aov_Chamberlins_2021,
    id_vars=['month'],  # Keep 'month' as the identifier
    value_vars=['Member AOV','Guest AOV'],  # Columns to melt
    var_name='Loyalty',  # Name for the new column containing year
    value_name='Avg Order Value'  # Name for the new column containing order volume
)

# Create the Plotly Express line chart
# Pass the correct DataFrame 'mon_aov_Chamberlins_2021_long'
fig = px.line(mon_aov_Chamberlins_2021_long, # Changed from mon_volume_Chamberlins_long
              x='month',
              y='Avg Order Value', # This column is now present
              color = 'Loyalty',
              title='Average Order Value by Seasons for loyalty status (Chamberlins, 2021)'
)

fig.show()
```

```
mon_aov_Chamberlins_2022_long = pd.melt(
    mon_aov_Chamberlins_2022,
    id_vars=['month'],  # Keep 'month' as the identifier
    value_vars=['Member AOV','Guest AOV'],  # Columns to melt
    var_name='Loyalty',  # Name for the new column containing year
    value_name='Avg Order Value'  # Name for the new column containing order volume
)

# Create the Plotly Express line chart
# Pass the correct DataFrame 'mon_aov_Chamberlins_2021_long'
fig2 = px.line(mon_aov_Chamberlins_2022_long, # Changed from mon_volume_Chamberlins_long
              x='month',
              y='Avg Order Value', # This column is now present
              color = 'Loyalty',
              title='Average Order Value by Seasons for loyalty status (Chamberlins, 2022)'
)

fig2.show()

mon_aov_Chamberlins_2023_long = pd.melt(
    mon_aov_Chamberlins_2023,
    id_vars=['month'],  # Keep 'month' as the identifier
    value_vars=['Member AOV','Guest AOV'],  # Columns to melt
    var_name='Loyalty',  # Name for the new column containing year
    value_name='Avg Order Value'  # Name for the new column containing order volume
)

# Create the Plotly Express line chart
# Pass the correct DataFrame 'mon_aov_Chamberlins_2021_long'
fig3 = px.line(mon_aov_Chamberlins_2023_long, # Changed from mon_volume_Chamberlins_long
              x='month',
              y='Avg Order Value', # This column is now present
              color = 'Loyalty',
              title='Average Order Value by Seasons for loyalty status (Chamberlins, 2023)'
)

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 observance of the average order value, loyalty memebrs always tend to purchase pricier goods than the guest customer.

Packaged Water sales by 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) &
```

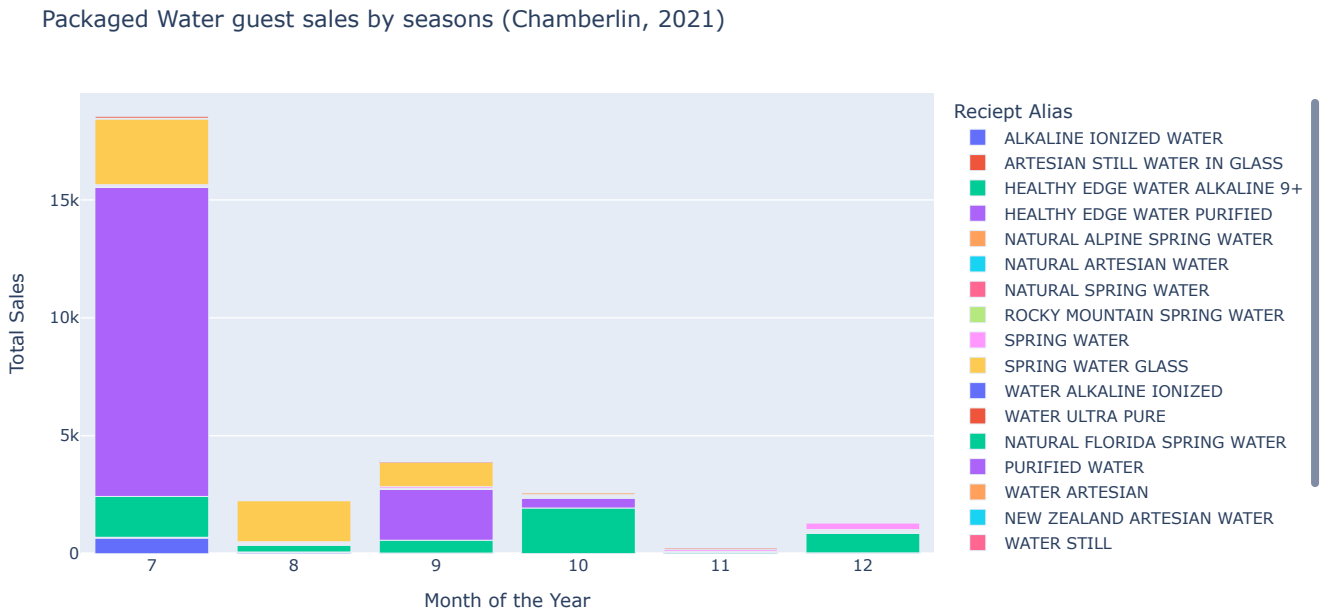
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

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

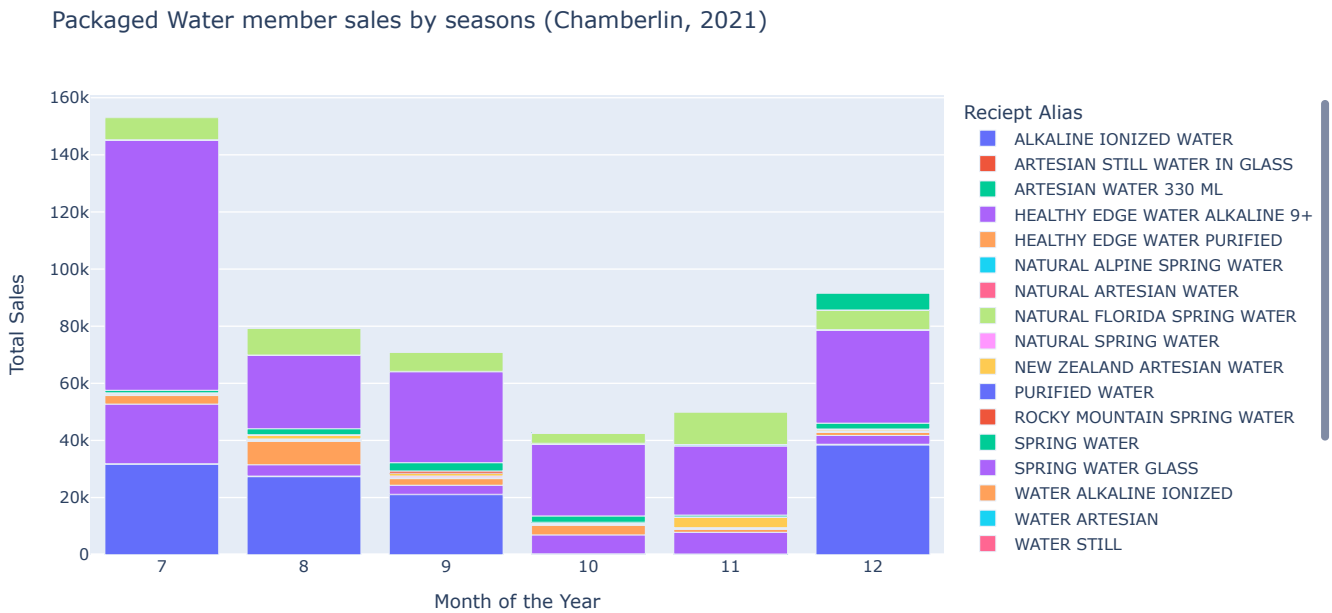
```
mon_sales_Chamberlins_member_packaged_water_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['Loyalty Customer?'] == 1) &
mon_sales_Chamberlins_member_packaged_water_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Loyalty Customer?'] == 1) &
mon_sales_Chamberlins_member_packaged_water_2023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





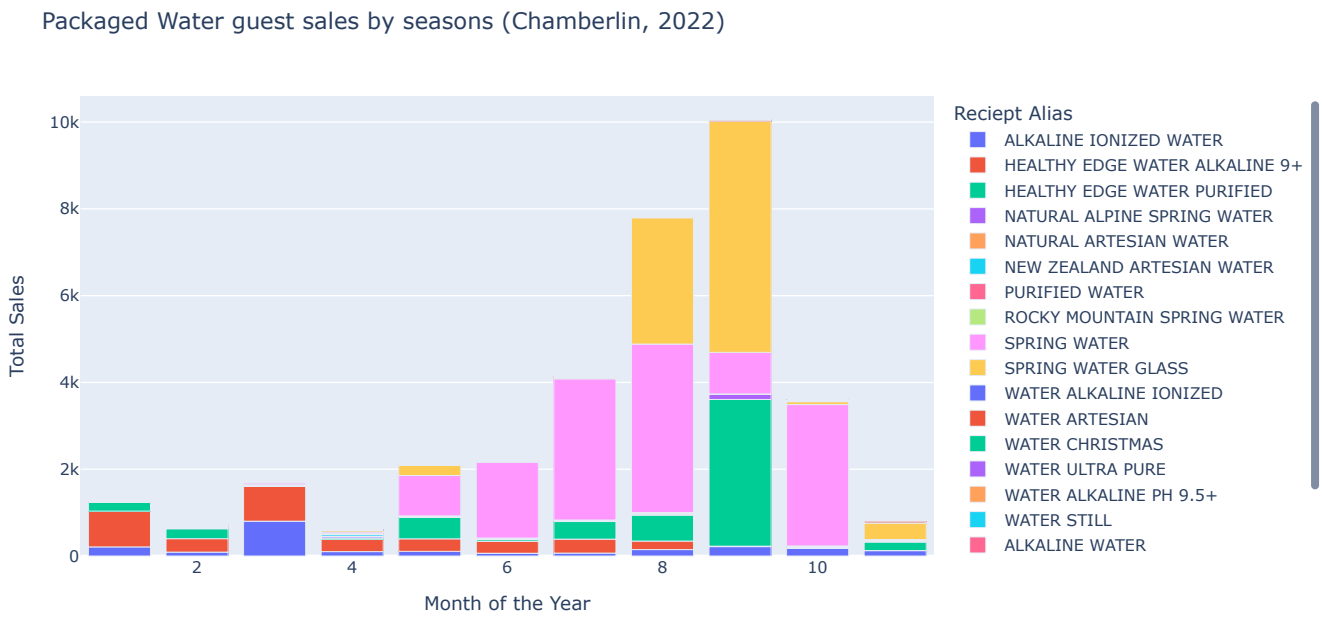
For guest customer, July's sales were mainly contributed by the Healthy Edge Purified Water, which covered the most of sales occured during the year. But the trend shift to Healthy Edge Alkatine taking over the most sold item with the following months.

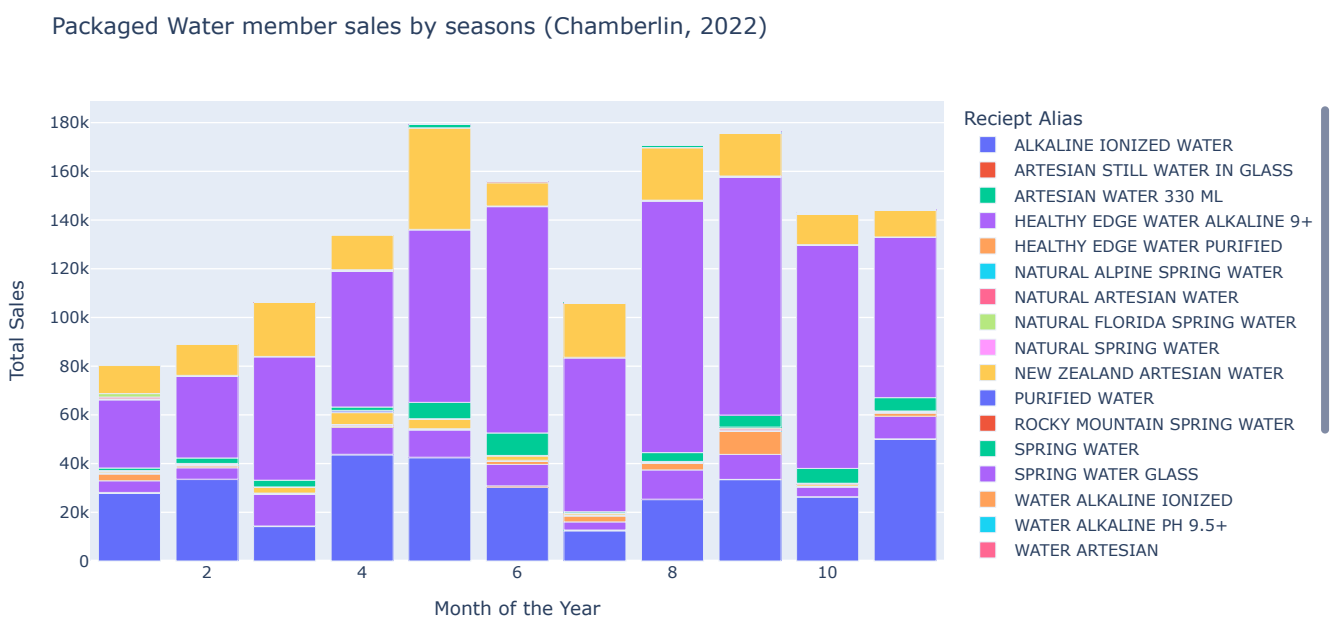
For loyalty customer, Spring Water Class was always the major part of the monthly sales, with Alkatine Ionized Water taking over some portions during the July, August, Septemebr and December. Again, the sales peak in month July.

```
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
```



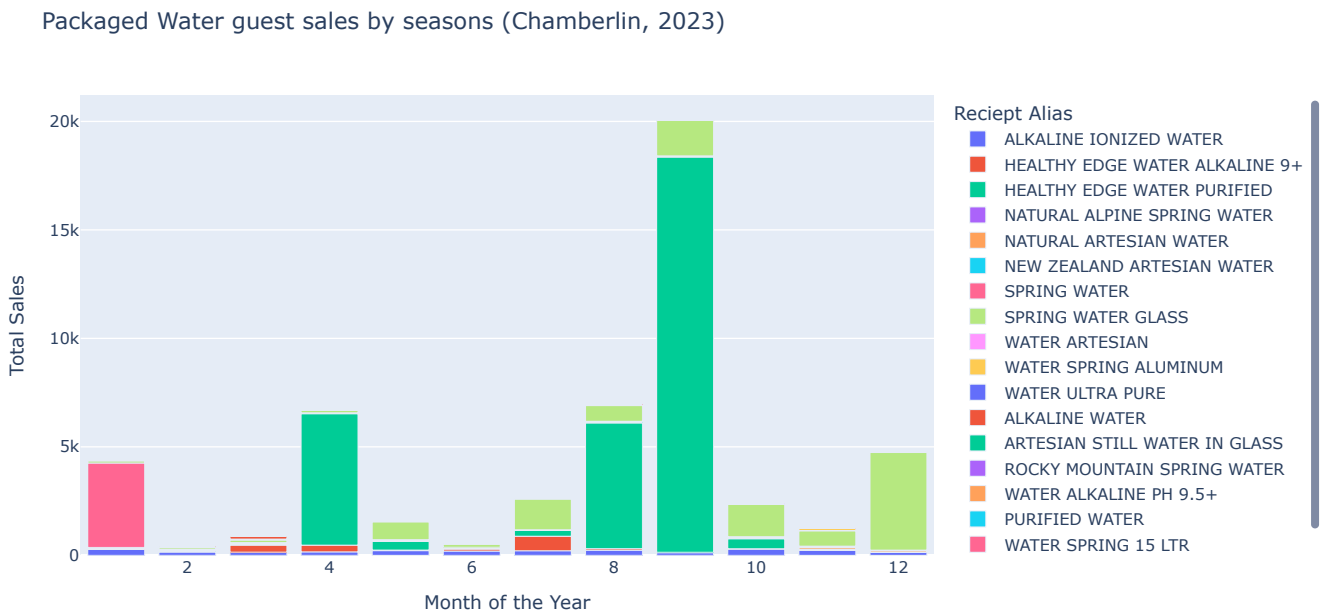


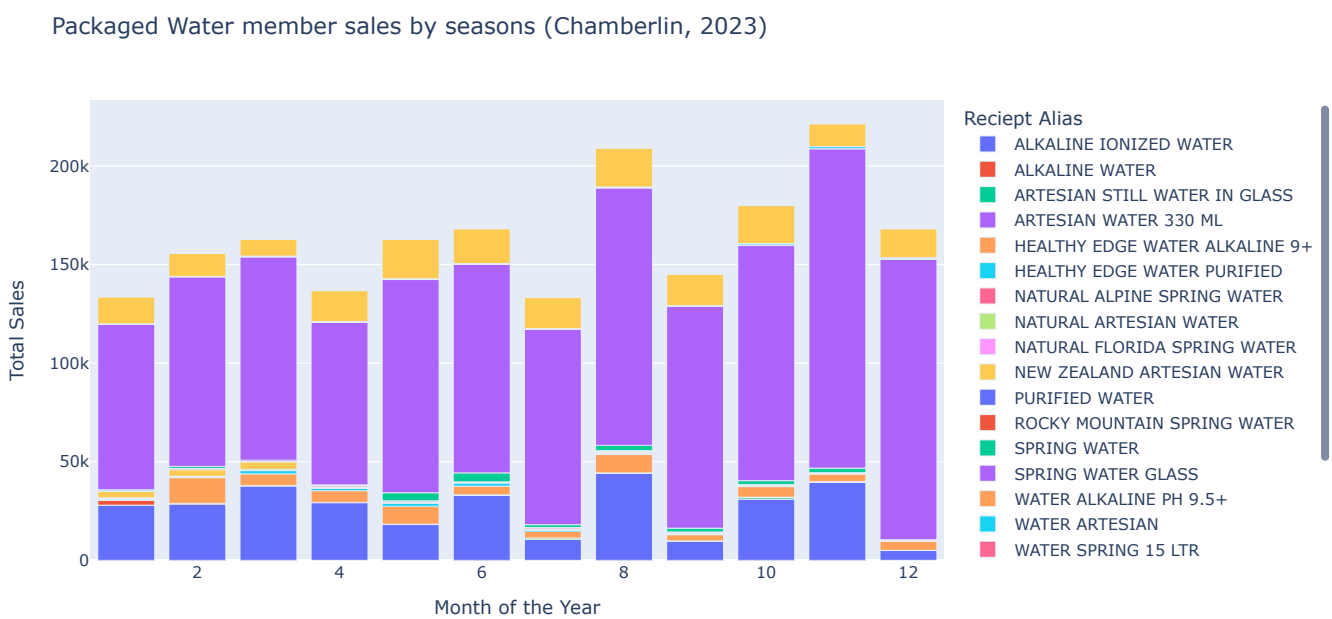
For guest customer, sales were attributed from Spring Water between May and October. It is worth noticing that in September, the sales for Spring Water Glass surged so high that it became the most sold item of the year for the guest group.

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.

Impact of promotions on product demand

As we delve further into the root cause of demand growth, we must probe on the imapct of promotional campaigns. We took the quantity sold as the target variable and would liek to performa root cause analysis of whether the scale of price discount would apply strong correlation with it.

```
#crete the discount scale column by using base price - selling price
basket_data_Chamberlins['price_gap'] = basket_data_Chamberlins['Base.Price'] - basket_data_Chamberlins['Selling.Price']
```

```
/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`

<ipython-input-316-84cbb9340602>:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view
```

```
Chamberlins_2022_discount = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2022].groupby(['Reciept Alias'])['price_g
```

```
/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`
```

```
Chamberlins_2022_WATER_Discount = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins['
Chamberlins_2022_WATER_Demand = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins['Re
Chamberlins_2023_WATER_Discount = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2023) & (basket_data_Chamberlins['
Chamberlins_2023_WATER_Demand = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2023) & (basket_data_Chamberlins['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`
```

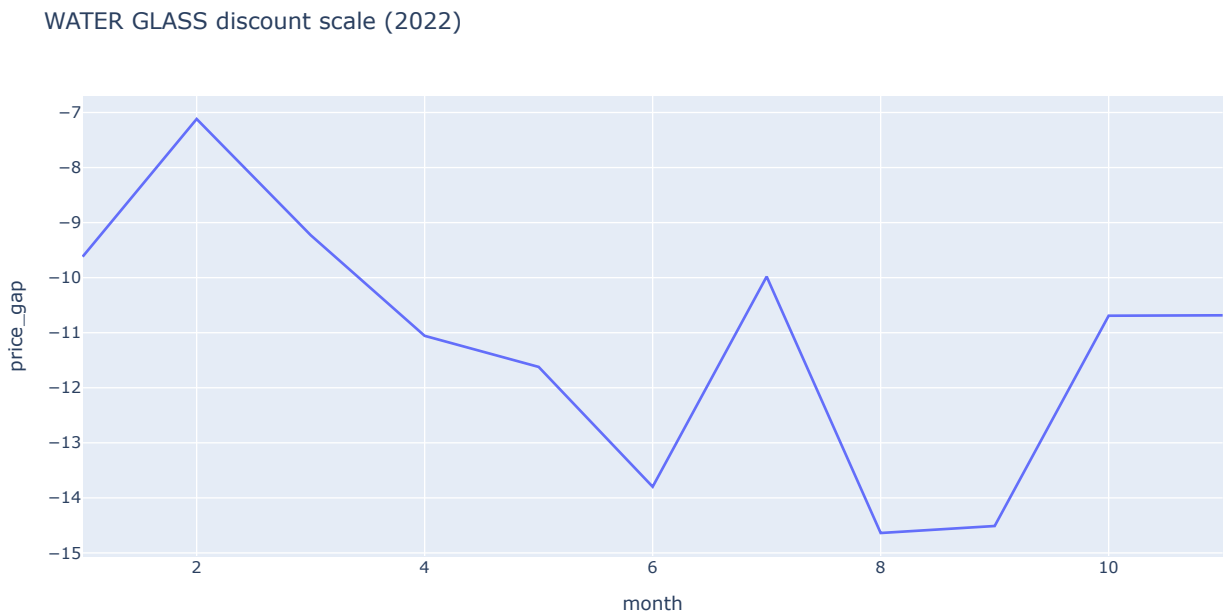
```
Chamberlins_2022_WATER_discount_sclae = Chamberlins_2022_WATER_Discount.merge(Chamberlins_2022_WATER_Demand, on = 'month')
Chamberlins_2023_WATER_discount_sclae = Chamberlins_2023_WATER_Discount.merge(Chamberlins_2023_WATER_Demand, on = 'month')
```

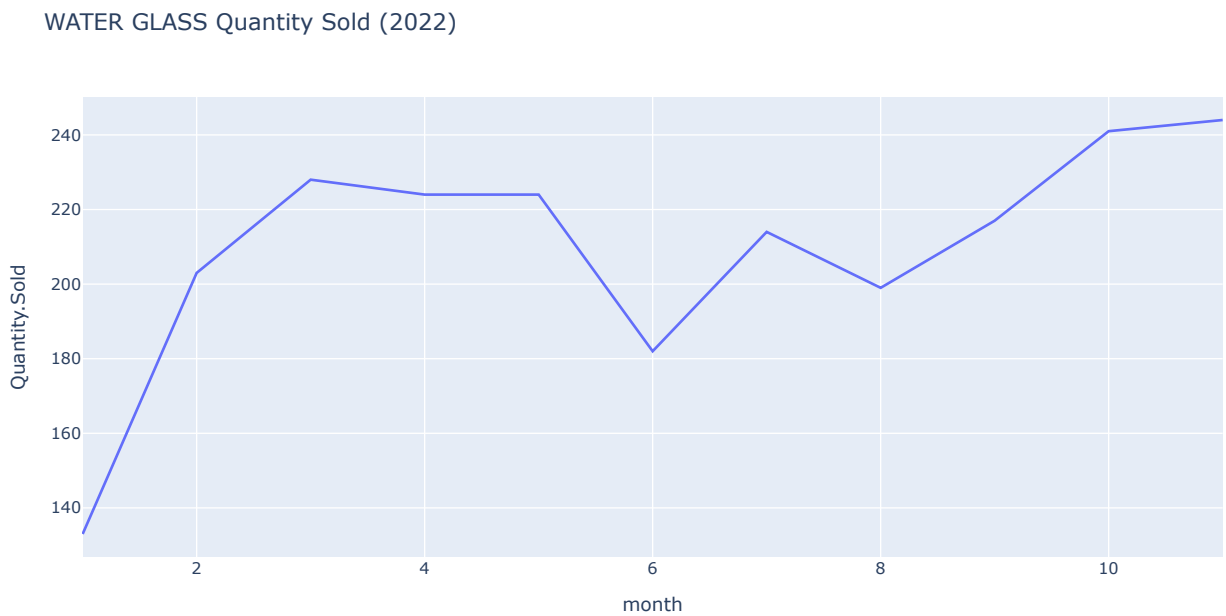
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

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

```
fig = px.line(Chamberlins_2022_WATER_discount_sclae,
              x = 'month',
              y='price_gap',
              title = 'WATER GLASS discount scale (2022)'

)
fig.show()

fig2 = px.line(Chamberlins_2022_WATER_discount_sclae,
              x = 'month',
              y='Quantity.Sold',
              title = 'WATER GLASS Quantity Sold (2022)'
```

```
)
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
```

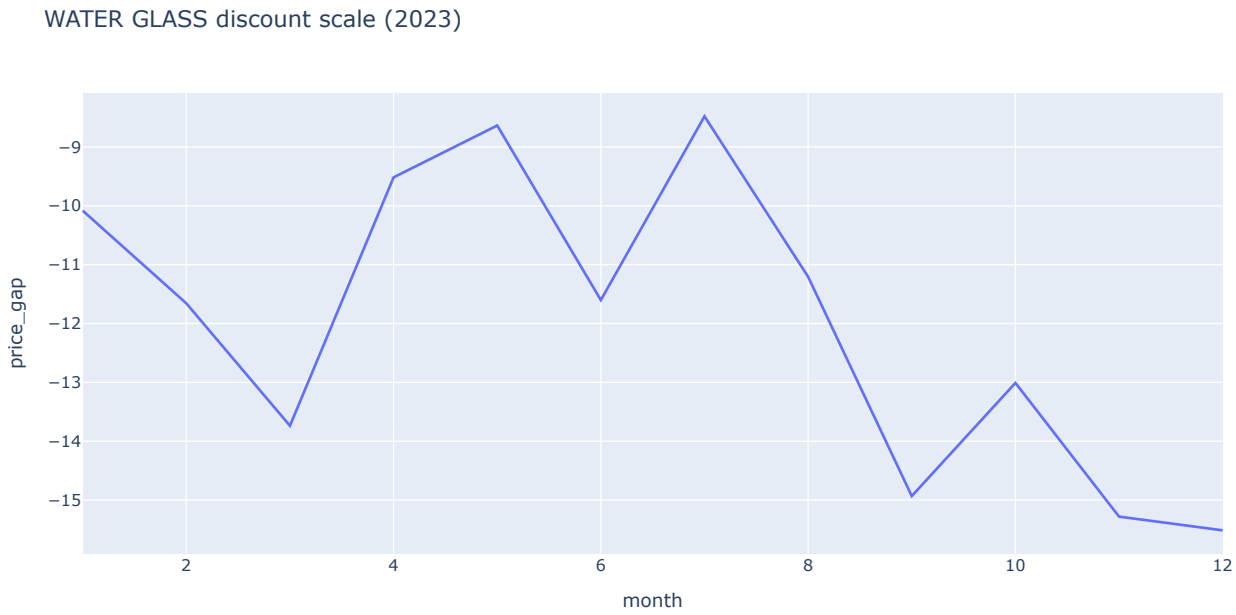


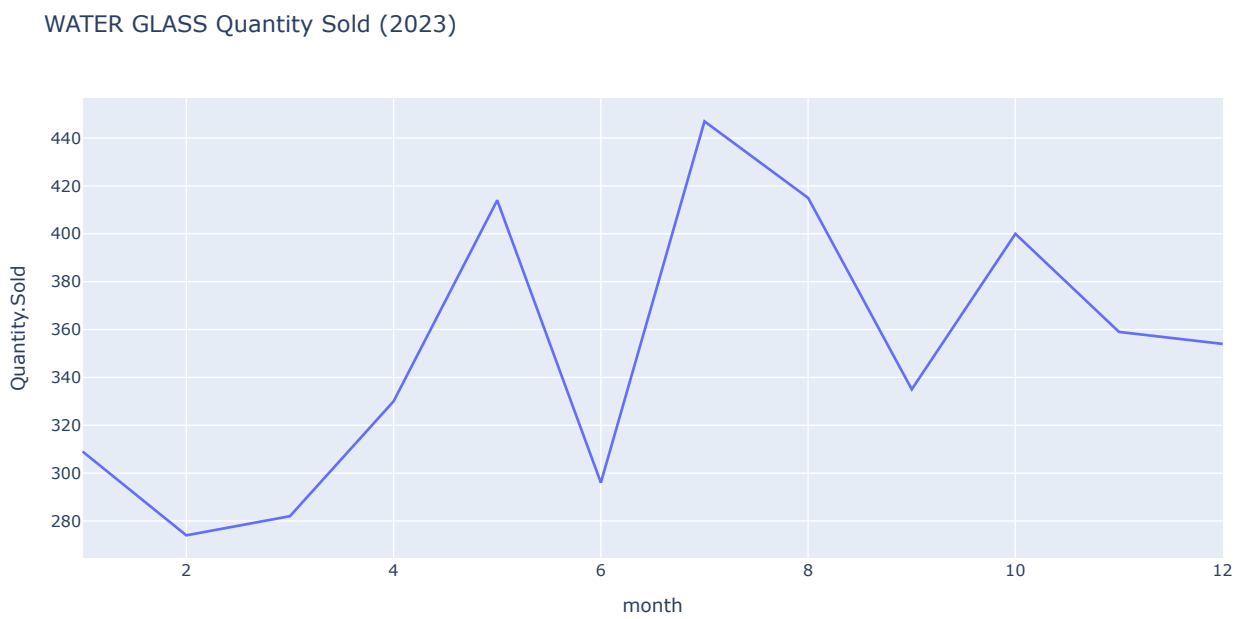



```
fig = px.line(Chamberlins_2023_WATER_discount_sclae,
              x = 'month',
              y='price_gap',
              title = 'WATER GLASS discount scale (2023)'

)
fig.show()

fig2 = px.line(Chamberlins_2023_WATER_discount_sclae,
              x = 'month',
              y='Quantity.Sold',
              title = 'WATER GLASS Quantity Sold (2023)'
```

```
)
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
```





In observance of the most popular product "Water Glass" as the example, we clarify that there exists storng correlation between the discoutned amount and the product demand. This valdiated our assupmtion of the promotion's impact: THe lower the price dropped will lead to higher demand, also making the item "GLass Water" a price-snesitive product.

Most popular 5 items from each loyalty tier

```
item_top_5 = basket_data_Chamberlins[basket_data_Chamberlins['year'] == 2021].groupby(['Reciept Alias', 'Loyalty Customer?'])['S
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
24435      SPRING WATER GLASS                  1  226805.6600
4874           CELERY OG EACH                  1  190059.0863
697    ALKALINE IONIZED WATER                  1  124145.6900
1622    AVOCADOS HASS OG EACH                  1  117102.4878
14937               LEMONS OG                  1   94146.9809
                        Reciept Alias  Loyalty Customer?     Sales
12558     HEALTHY EDGE WATER PURIFIED                  0  15749.35
4873                   CELERY OG EACH                  0   6402.60
24434              SPRING WATER GLASS                  0   5585.22
12556  HEALTHY EDGE WATER ALKALINE 9+                  0   5369.48
1621            AVOCADOS HASS OG EACH                  0   4263.29
```

Both water and veggies are the top sellers for members and guests. The nuances lay within 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 Avocado 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`
```

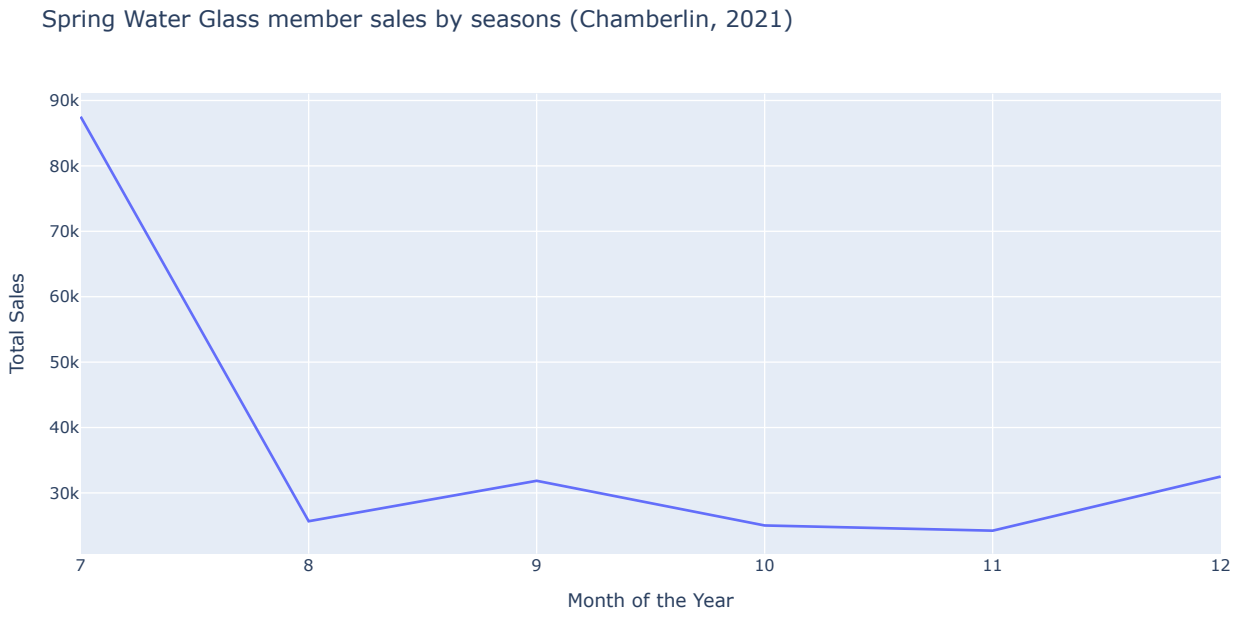
```
basket_data_Chamberlins_guest_HE_Purified_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'HEALTHY E
basket_data_Chamberlins_guest_HE_Purified_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'HEALTHY E
basket_data_Chamberlins_guest_HE_Purified_2023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

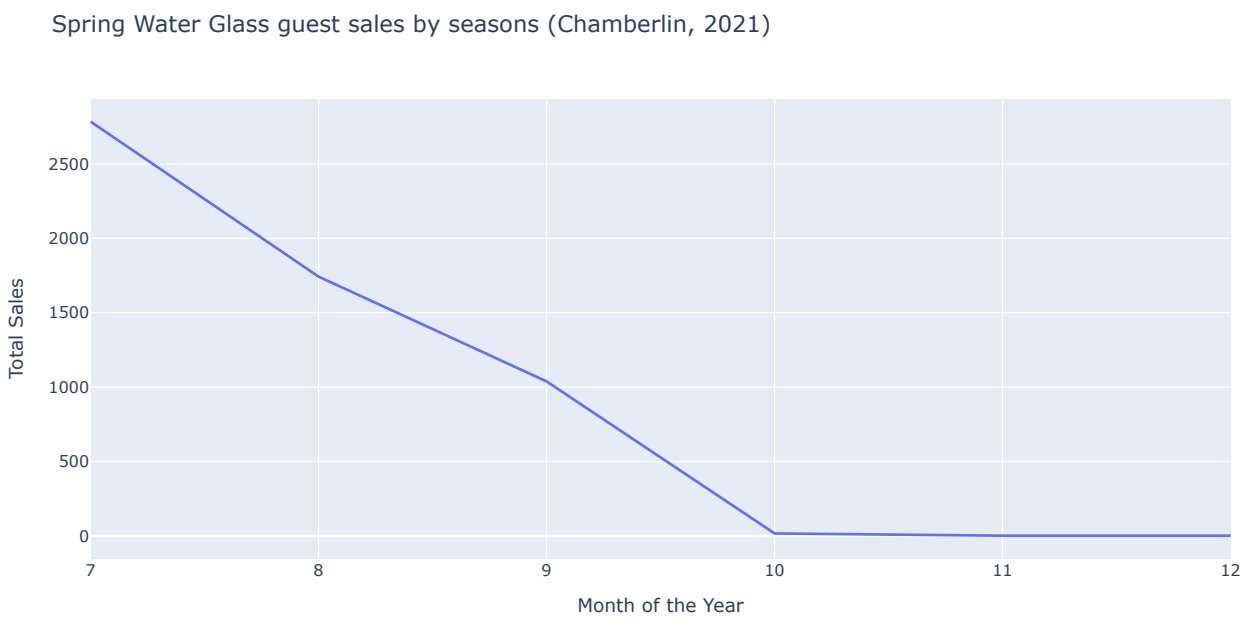
```
/usr/local/lib/python3.11/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

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

```
fig = px.line(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.line(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.line(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
```

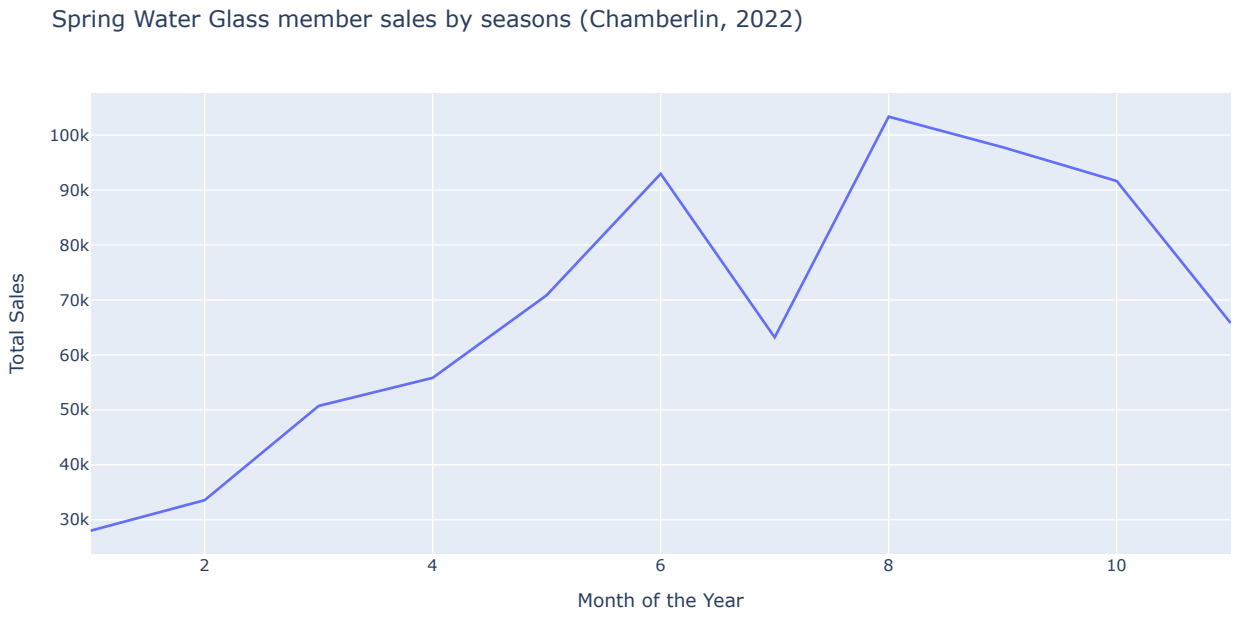


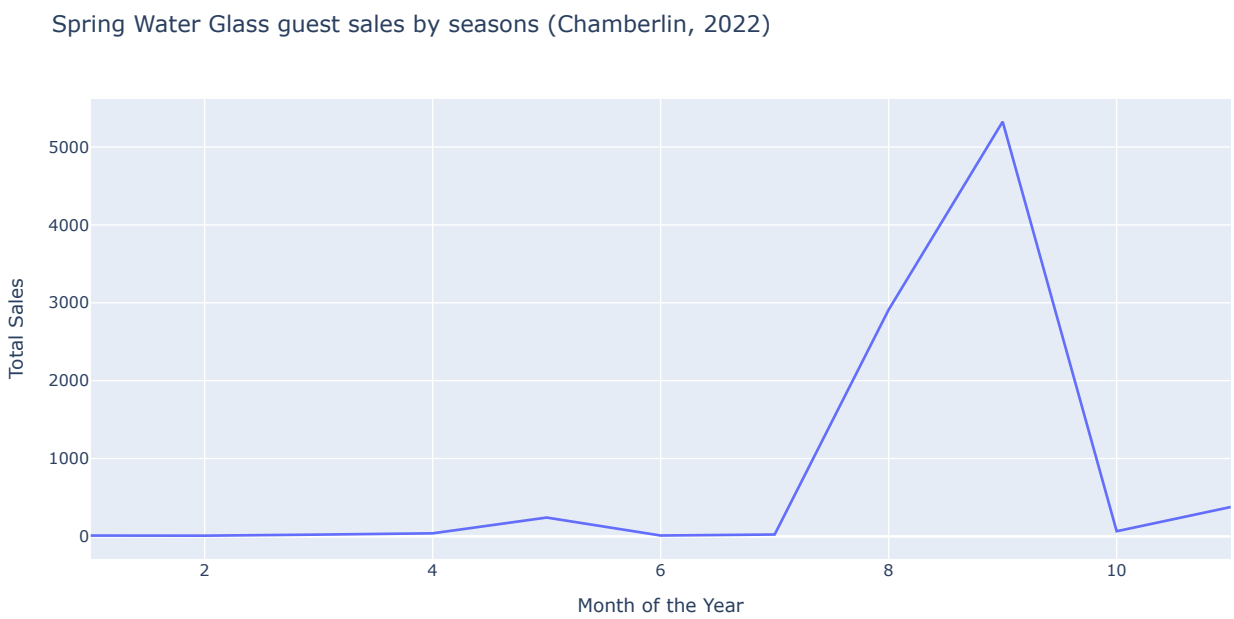


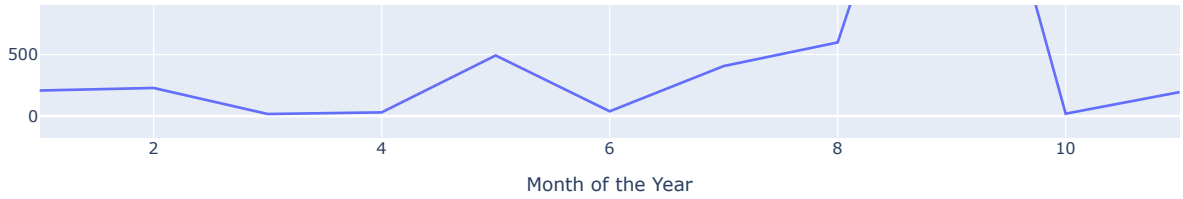
Both sales for Spring awter Glass and Healthy Edge Purified Water steadily decrease over time. In year 2021, the sales of H.E. Purified Water significantly surpass that of the Spring Water Glass within guest group. This could mean that the price of the Healthy Edge Purified water can be significantly lower than that of the Spring Glass.

```
fig = px.line(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.line(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.line(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/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, both sales icnrement from month to month and peaked in 3rd quarter. THe sales spike in September could mark some potential sales campaigns that drived up intuition of guest purchasing.

Average Sales Value for each loyalty tiers

```
basket_data_Chamberlins.columns
```

```
/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`

Index(['Customer.Total.Proper', 'Transaction.End.Date.Proper', 'Quantity.Sold',
       'Base.Price', 'Selling.Price', 'Department Name Transaction', 'month',
       'day', 'year', 'Transaction Store', 'Banner', 'Item ID',
       'Reciept Alias', 'Item Size', 'Brand Name', 'Item Type', 'Date Created',
       'Category', 'Subcategory', 'Anonymous Customer Number',
       'Loyalty Customer?', 'Opted Into Marketing', 'Loyalty Balance',
       'Discount receiving?', 'Customer Number', 'Receipt Number',
       'Employee Number', 'Department Number', 'Sales', 'price_gap'],
      dtype='object')
```

```
mon_aov_member_Chamberlins_2021 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2021) & (basket_data_Chamberlins['
mon_aov_member_Chamberlins_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins['
mon_aov_member_Chamberlins_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`
```

Key profit driver: Price or Demand?

When we further break down the components of total sales, we must wonder if the sales uprise is attributed from the sold quantities or from a higher markup. In this part of the analysis I tried to examinine which part was identified as the actual sale driver. I started with speculating the unit price of the product 'Healthy Edge Purified Water' to see the actual correlation:

```
# prompt: find all outlier of selling prices from basket_data_Chamberlins year 2022


# Assuming 'basket_data_Chamberlins' is your DataFrame
# and it has columns 'year', 'Sales'

# Filter data for the year 2022
sales_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['year'] == 2022) & (basket_data_Chamberlins['Reciept Alias'] == 'S

# Calculate quartiles
Q1 = sales_2022.quantile(0.25)
Q3 = sales_2022.quantile(0.75)
IQR = Q3 - Q1

# Define bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Identify outliers
outliers = sales_2022[(sales_2022 < lower_bound) | (sales_2022 > upper_bound)]

# Print or further process the outliers
```

```
print("Outliers in selling prices for 2022:")
sales_2022
```

```
/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`

Outliers in selling prices for 2022:
```

	Selling.Price
0	2.99

dtype: float64

```
#Speculate the price difference for products between two loyalty tiers:
basket_data_Chamberlins_guest_HE_Purified_Price_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'HEA
basket_data_Chamberlins_member_HE_Purified_Price_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'HE
basket_data_Chamberlins_member_Spring_Price_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRING
basket_data_Chamberlins_guest_Spring_Price_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRING W
```

```
/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_Price_min_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] ==
basket_data_Chamberlins_member_HE_Purified_Price_min_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] ==
basket_data_Chamberlins_member_Spring_Price_min_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPR
basket_data_Chamberlins_guest_Spring_Price_min_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRI
```

```
/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_Price_max_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] ==
basket_data_Chamberlins_member_HE_Purified_Price_max_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] ==
basket_data_Chamberlins_member_Spring_Price_max_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPR
basket_data_Chamberlins_guest_Spring_Price_max_2022 = basket_data_Chamberlins[(basket_data_Chamberlins['Reciept Alias'] == 'SPRI
```

```
/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_Spring_Price_2022.merge(basket_data_Chamberlins_guest_Spring_Price_max_2022, on='month',how='outer
```

```
/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	Selling.Price_x	Selling.Price_y
0	1	1.99	2.79
1	2	1.99	1.99
2	3	2.29	2.29
3	4	2.79	5.58
4	5	1.99	22.32
5	6	1.99	2.79
6	7	1.99	2.50
7	8	1.99	71.64
8	9	1.99	95.52
9	10	1.99	2.99
10	11	1.99	23.88

```
basket_data_Chamberlins_member_Spring_Price_2022 = basket_data_Chamberlins_member_Spring_Price_2022.merge(basket_data_Chamberlin
basket_data_Chamberlins_guest_Spring_Price_2022 = basket_data_Chamberlins_guest_Spring_Price_2022.merge(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`
```

```
basket_data_Chamberlins_guest_Spring_Price_2022
```

```
/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`
```

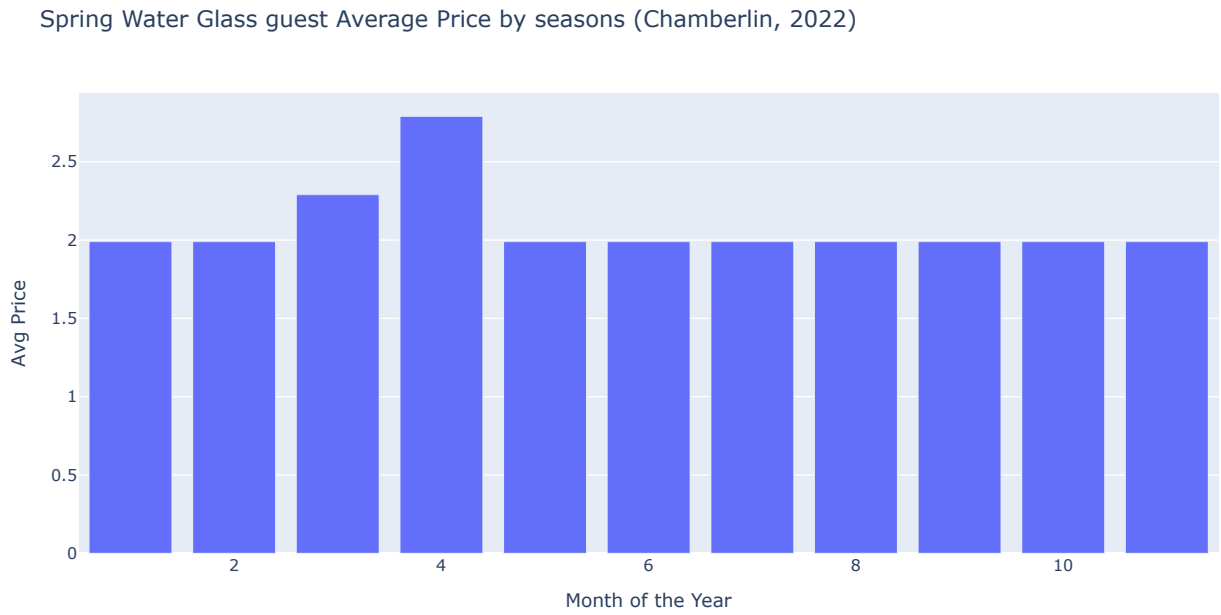
	month	Avg Price	Max Price	Min Price
0	1	1.99	2.79	1.99
1	2	1.99	1.99	1.79
2	3	2.29	2.29	1.49
3	4	2.79	5.58	1.79
4	5	1.99	22.32	1.99
5	6	1.99	2.79	1.99
6	7	1.99	2.50	1.50
7	8	1.99	71.64	1.99
8	9	1.99	95.52	1.99
9	10	1.99	2.99	1.50
10	11	1.99	23.88	1.99

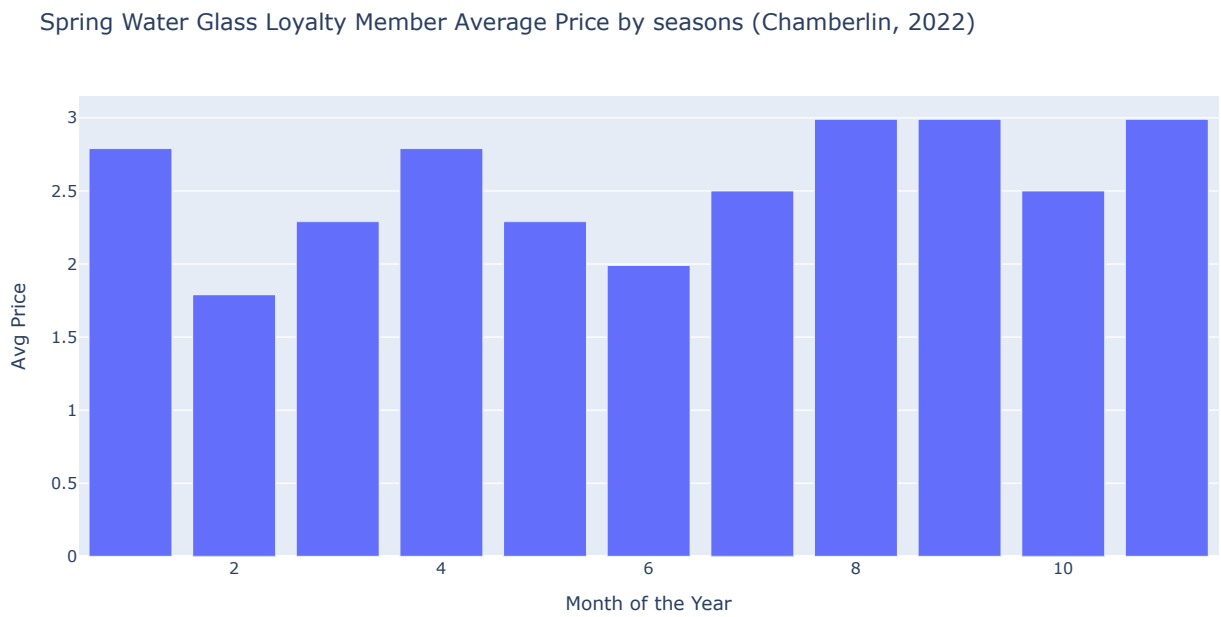
basket_data_Chamberlins_guest_Spring_Price_2022 | New interactive sheet

```
fig = px.bar(basket_data_Chamberlins_guest_Spring_Price_2022,
             x = 'month',
             y= 'Avg Price',
             title = 'Spring Water Glass guest Average Price by seasons (Chamberlin, 2022)',
             labels={'month': 'Month of the Year'},
             barmode='group')
fig.show()

fig2 = px.bar(basket_data_Chamberlins_member_Spring_Price_2022,
```

```
             x = 'month',
             y= 'Avg Price',
             title = 'Spring Water Glass Loyalty Member Average Price by seasons (Chamberlin, 2022)',
             labels={'month': 'Month of the Year'},
             barmode='group')
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
```

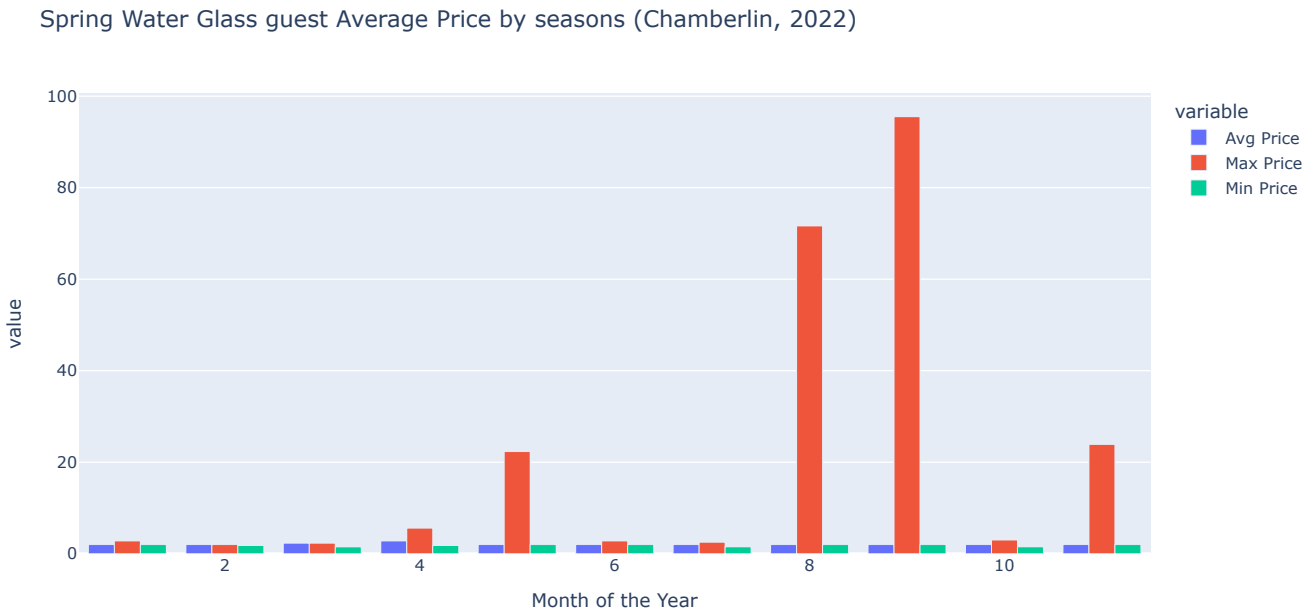





```
fig = px.bar(basket_data_Chamberlins_guest_Spring_Price_2022,
             x = 'month',
             y= ['Avg Price','Max Price','Min Price'],
             title = 'Spring Water Glass guest Average Price by seasons (Chamberlin, 2022)',
             labels={'month': 'Month of the Year'},
             barmode='group')
fig.show()
```

```
fig2 = px.bar(basket_data_Chamberlins_member_Spring_Price_2022,
             x = 'month',
             y= ['Avg Price','Max Price','Min Price'],
             title = 'Spring Water Glass Loyalty Member Average Price by seasons (Chamberlin, 2022)',
             labels={'month': 'Month of the Year'},
             barmode='group')
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
```



In observance of a couple outliers for maximum price paid for the Spring Water, we can suspect that some abnormal purchasing behaviors occured during the year of 2022. It can either be due to the Covid supply shortage which caused certain item to rise up in the price, or simply an inputing error to remain for further probing.

```
fig = px.bar(basket_data_Chamberlins_guest_HE_Purified_Price_2022,
             x = 'month',
```