In [1]:

```
# Import Libraries for the Project
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import numpy as np
from collections import Counter
from scipy.stats import skew, kurtosis
pd.set_option('display.max_columns', None)
plt.style.use('ggplot')
import plotly.express as px
import squarify
```

In [2]:
```
df = pd.read_excel(r"C:\Users\HENRY OKEOMA\Downloads\Sterling E-Commerce Data.xlsx")
df
```

Out[2]:

	Category	City	County	Cust Id	Customer Since	Date of Order	Full Name	Gender	Item Id	Order Id	Payment Method	Place Name	N
0	Health & Sports	Bode	Humboldt	112285	2008-02-11	2022-08-07	Renaud, Maudie	F	880913	100547952.0	Easypay_MA	Bode	352
1	Men's Fashion	Belleville	St. Clair	112386	2005-06-23	2022-08-08	Shimp, Mariela	F	881493	100548328.0	Easypay_MA	Belleville	310
2	Men's Fashion	Belleville	St. Clair	112386	2005-06-23	2022-08-08	Shimp, Mariela	F	881492	100548328.0	Easypay_MA	Belleville	310
3	Computing	Young America	Carver	112501	2013-09-15	2022-08-18	Doiron, Latrina	F	886067	100551079.0	Payaxis	Young America	578
4	Entertainment	Young America	Carver	112501	2013-09-15	2022-08-20	Doiron, Latrina	F	886878	100551618.0	Payaxis	Young America	578
...	...	...	...	...	...	...	...	...	...	...	...	...	
283078	Women's Fashion	Burkettsville	Mercer	81251	2013-10-15	2021-12-30	Kester, Apolonia	F	700522	100428972.0	cod	Burkettsville	572
283079	Women's Fashion	Burkettsville	Mercer	81251	2013-10-15	2021-12-30	Kester, Apolonia	F	700518	100428972.0	cod	Burkettsville	572
283080	Women's Fashion	Burkettsville	Mercer	81251	2013-10-15	2021-12-30	Kester, Apolonia	F	700520	100428972.0	cod	Burkettsville	572
283081	Women's Fashion	Burkettsville	Mercer	81251	2013-10-15	2021-12-30	Kester, Apolonia	F	700517	100428972.0	cod	Burkettsville	572
283082	Women's Fashion	Burkettsville	Mercer	81251	2013-10-15	2021-12-30	Kester, Apolonia	F	700519	100428972.0	cod	Burkettsville	572

283083 rows × 19 columns

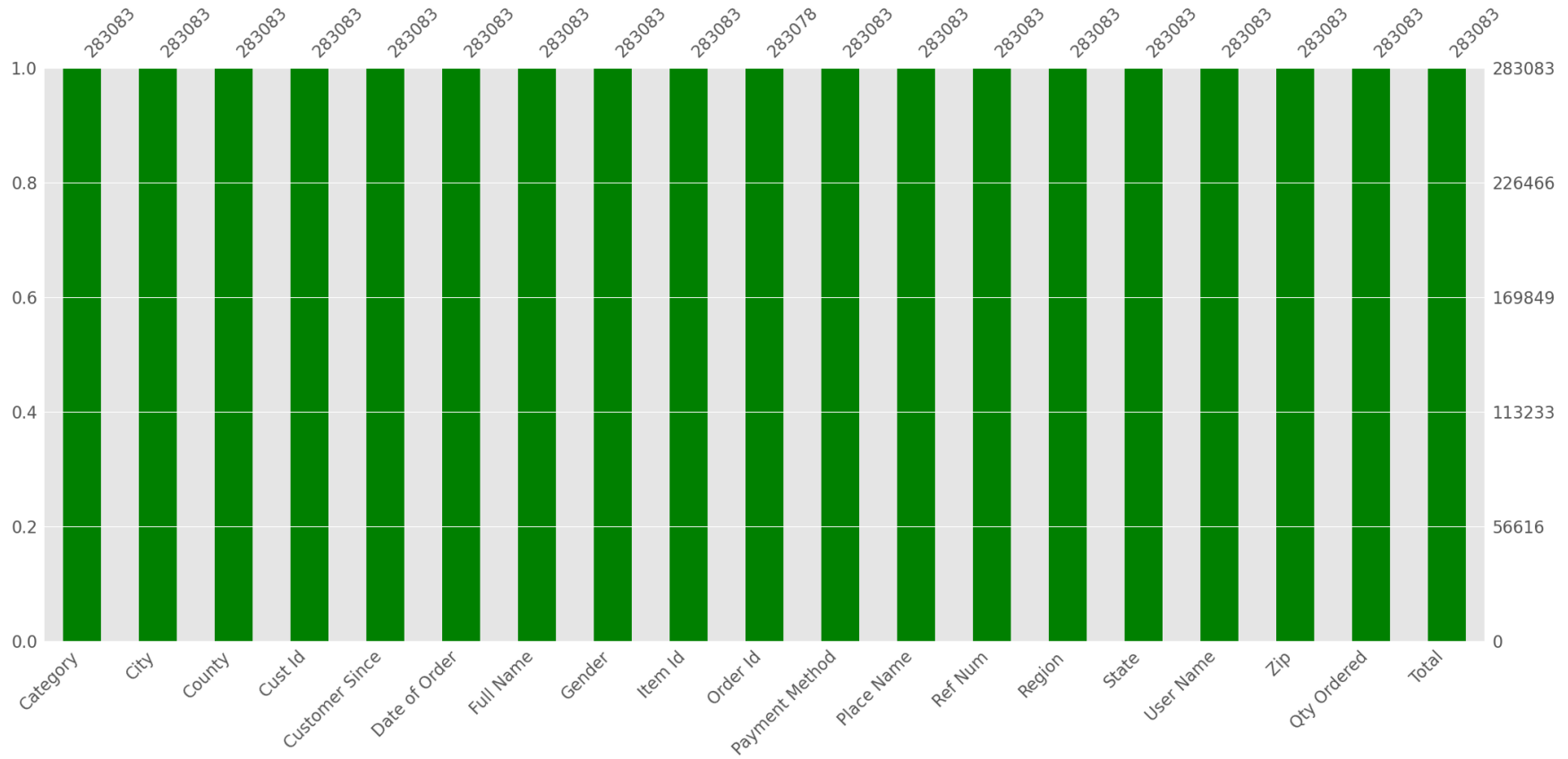
In [3]:

```
# Check data for more understanding data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 283083 entries, 0 to 283082
Data columns (total 19 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Category        283083 non-null  object
 1   City            283083 non-null  object
 2   County          283083 non-null  object
 3   Cust Id         283083 non-null  int64
 4   Customer Since  283083 non-null  datetime64[ns]
 5   Date of Order   283083 non-null  datetime64[ns]
 6   Full Name       283083 non-null  object
 7   Gender          283083 non-null  object
 8   Item Id         283083 non-null  int64
 9   Order Id        283078 non-null  float64
 10  Payment Method  283083 non-null  object
 11  Place Name      283083 non-null  object
 12  Ref Num         283083 non-null  int64
 13  Region          283083 non-null  object
 14  State           283083 non-null  object
 15  User Name       283083 non-null  object
 16  Zip             283083 non-null  int64
 17  Qty Ordered     283083 non-null  int64
 18  Total           283083 non-null  float64
dtypes: datetime64[ns](2), float64(2), int64(5), object(10)
memory usage: 41.0+ MB
```

We have 10 Objects (Categories), 5 integers and 2 floats on our data set. Also we can sport missing value in the order ID, we shall visualise our missing value

In [4]:
```
msno.bar(df, color="green");
```



Our order ID have 5 missing values also see the below details

In [5]:
```
df.isna().sum()
```

Out[5]:
```
Category          0
City              0
County            0
Cust Id           0
Customer Since    0
Date of Order     0
Full Name         0
Gender            0
Item Id           0
Order Id          5
Payment Method    0
Place Name        0
Ref Num           0
Region            0
State             0
User Name         0
Zip               0
Qty Ordered       0
Total             0
dtype: int64
```

We have only five missing value in our dataset

In [6]:
```
# Description of the Data Frame
df.describe().astype(int)
```

Out[6]:

	Cust Id	Item Id	Order Id	Ref Num	Zip	Qty Ordered	Total
count	283083	283083	283078	283083	283083	283083	283083
mean	70106	741747	100456970	561107	49147	3	816
std	30215	95664	60909	256101	27235	4	1986
min	4	574769	100354677	111127	210	1	0
25%	56640	659898	100404736	341071	26264	2	49
50%	74320	742471	100451836	565623	48808	2	149
75%	92371	826078	100513392	782211	72004	3	800
max	115326	905208	100562387	999981	99402	501	101262

In [7]:
```
# Checking for Duplicates
df.duplicated().sum()
```

Out[7]:
```
0
```

We have no duplicates in our dataset.

In [8]:
```
# Dropping the missing Value
df.dropna(subset=['Order Id'],inplace=True)
df.isna().sum()
```

Out[8]:
```
Category          0
City              0
County            0
Cust Id           0
Customer Since    0
Date of Order     0
Full Name         0
Gender            0
Item Id           0
Order Id          0
Payment Method    0
Place Name        0
Ref Num           0
Region            0
State             0
User Name         0
Zip               0
Qty Ordered       0
Total             0
dtype: int64
```

We have dropped all the missing values and our data set is almost ready for visualisation

In [9]:
```
df.columns
```

Out[9]:
```
Index(['Category', 'City', 'County', 'Cust Id', 'Customer Since',
       'Date of Order', 'Full Name', 'Gender', 'Item Id', 'Order Id',
       'Payment Method', 'Place Name', 'Ref Num', 'Region', 'State',
       'User Name', 'Zip', 'Qty Ordered', 'Total'],
      dtype='object')
```

In [10]:
```
# Datetime Analysis
df.head(2)
```

Out[10]:

	Category	City	County	Cust Id	Customer Since	Date of Order	Full Name	Gender	Item Id	Order Id	Payment Method	Place Name	Ref Num	Region	S
0	Health & Sports	Bode	Humboldt	112285	2008-02-11	2022-08-07	Renaud, Maudie	F	880913	100547952.0	Easypay_MA	Bode	352808	Midwest	
1	Men's Fashion	Belleville	St. Clair	112386	2005-06-23	2022-08-08	Shimp, Mariela	F	881493	100548328.0	Easypay_MA	Belleville	310849	Midwest	

In [11]:
```
# Modifying our Columns names using the camel case style
df.columns = ['Category', 'City', 'County', 'CustId', 'CustomerSince',
       'Date', 'FullName', 'Gender', 'ItemId', 'OrderId',
       'PaymentMethod', 'PlaceName', 'RefNum', 'Region', 'State',
       'UserName', 'Zip', 'QtyOrdered', 'Total']
df.head(2)
```

Out[11]:

	Category	City	County	CustId	CustomerSince	Date	FullName	Gender	ItemId	OrderId	PaymentMethod	PlaceName	RefNu
0	Health & Sports	Bode	Humboldt	112285	2008-02-11	2022-08-07	Renaud, Maudie	F	880913	100547952.0	Easypay_MA	Bode	3528(
1	Men's Fashion	Belleville	St. Clair	112386	2005-06-23	2022-08-08	Shimp, Mariela	F	881493	100548328.0	Easypay_MA	Belleville	3108

In [12]:
```
# Extract the Year, Month and Quater from Date, Also extract the First year the customer Joined
df['Year_1st_order'] = df['CustomerSince'].dt.year
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Month_name'] = df['Date'].dt.month_name()
df['Day_name'] = df['Date'].dt.day_name()
df['Quarter'] = df['Date'].dt.quarter


df.head(2)
```

Out[12]:

	Category	City	County	CustId	CustomerSince	Date	FullName	Gender	ItemId	OrderId	PaymentMethod	PlaceName	RefNu
0	Health & Sports	Bode	Humboldt	112285	2008-02-11	2022-08-07	Renaud, Maudie	F	880913	100547952.0	Easypay_MA	Bode	3528(
1	Men's Fashion	Belleville	St. Clair	112386	2005-06-23	2022-08-08	Shimp, Mariela	F	881493	100548328.0	Easypay_MA	Belleville	3108

In [13]:
```
df.columns
```

Out[13]:
```
Index(['Category', 'City', 'County', 'CustId', 'CustomerSince', 'Date',
       'FullName', 'Gender', 'ItemId', 'OrderId', 'PaymentMethod', 'PlaceName',
       'RefNum', 'Region', 'State', 'UserName', 'Zip', 'QtyOrdered', 'Total',
       'Year_1st_order', 'Year', 'Month', 'Month_name', 'Day_name', 'Quarter'],
      dtype='object')
```

In [30]:

```
# Create a Function to determine Customer years and loyalty and view the 3 rows
def Cus_years(x):
    if x >= 2016:
        return 'Newest_Cus_6'
    elif x >= 2010:
        return 'Recent_Cus_12'
    elif x >= 2000:
        return 'GenZ_Cus_22'
    elif x >= 1989:
        return 'Adult_Cus_33'
    else:
        return 'Advanced_Cus_44'
# Apply Function to the data
df['Cus_years_Loyalty'] = df['Year_1st_order'].apply(Cus_years)

df.head(3)
```

Out[30]:

ion	State	UserName	Zip	QtyOrdered	Total	Year_1st_order	Year	Month	Month_name	Day_name	Quarter	Cus_years	Season	Cus_
est	IA	mcrenaud	50519	3	32.0	2008	2022	8	August	Sunday	3	GenZ_Cus_22	Summer	
est	IL	mgshimp	62223	2	74.8	2005	2022	8	August	Monday	3	GenZ_Cus_22	Summer	
est	IL	mgshimp	62223	2	74.9	2005	2022	8	August	Monday	3	GenZ_Cus_22	Summer	

We have created a new Column to determine loyalty of the customer based on the number of years since their first order

In [31]:

```
# Also lets create another another Function and a Column to define season
def Season(x):
    if x == 'March':
        return 'Spring'
    elif x == 'April':
        return 'Spring'
    elif x == 'April':
        return 'Spring'
    elif x == 'May':
        return 'Spring'
    elif x ==  'June':
        return 'Summer'
    elif x ==  'July':
        return 'Summer'
    elif x ==  'August':
        return 'Summer'
    elif x == 'September':
        return 'Autumn'
    elif x == 'October':
        return 'Autumn'
    elif x == 'November':
        return 'Autumn'
    elif x == 'December':
        return 'Winter'
    elif x == 'January':
        return 'Winter'
    elif x == 'February':
        return 'Winter'

df['Season'] = df['Month_name'].apply(Season)
df.head(3)
```

Out[31]:

	Category	City	County	CustId	CustomerSince	Date	FullName	Gender	ItemId	OrderId	PaymentMethod	PlaceName	RefNu
0	Health & Sports	Bode	Humboldt	112285	2008-02-11	2022-08-07	Renaud, Maudie	F	880913	100547952.0	Easypay_MA	Bode	3528(
1	Men's Fashion	Belleville	St. Clair	112386	2005-06-23	2022-08-08	Shimp, Mariela	F	881493	100548328.0	Easypay_MA	Belleville	3108
2	Men's Fashion	Belleville	St. Clair	112386	2005-06-23	2022-08-08	Shimp, Mariela	F	881492	100548328.0	Easypay_MA	Belleville	3108

In [32]:
```
# We have some data points with Total amount paid  = Zero
df[df['Total'] == 0]
```

Out[32]:

	Category	City	County	CustId	CustomerSince	Date	FullName	Gender	ItemId	OrderId	PaymentMethod	PlaceName	RefNum	Region	S

We have about 18924 rows with totals as zero, we shall replace this with the median values

In [34]:
```
# We shall replace the 0.0 values in our total column with NaN and
#further go ahead to replace with the median of each category
df.replace(0.0, np.nan, inplace=True)
```

In [35]:
```
# We then have to replace the NaN values with the median grouping by category

df['Total'] = df.groupby('Category')['Total'].transform(lambda x: x.fillna(x.median()))
```

In [36]:
```
# Checking if we still have value as 0.0 in the Total Column
df[df['Total'] == 0]
```

Out[36]:

	Category	City	County	CustId	CustomerSince	Date	FullName	Gender	ItemId	OrderId	PaymentMethod	PlaceName	RefNum	Region	S

In [37]:
```
df.head(2)
```

Out[37]:

	Category	City	County	CustId	CustomerSince	Date	FullName	Gender	ItemId	OrderId	PaymentMethod	PlaceName	RefNu
0	Health & Sports	Bode	Humboldt	112285	2008-02-11	2022-08-07	Renaud, Maudie	F	880913	100547952.0	Easypay_MA	Bode	3528(
1	Men's Fashion	Belleville	St. Clair	112386	2005-06-23	2022-08-08	Shimp, Mariela	F	881493	100548328.0	Easypay_MA	Belleville	3108

Our Dataset is clean and ready for all Visualisations

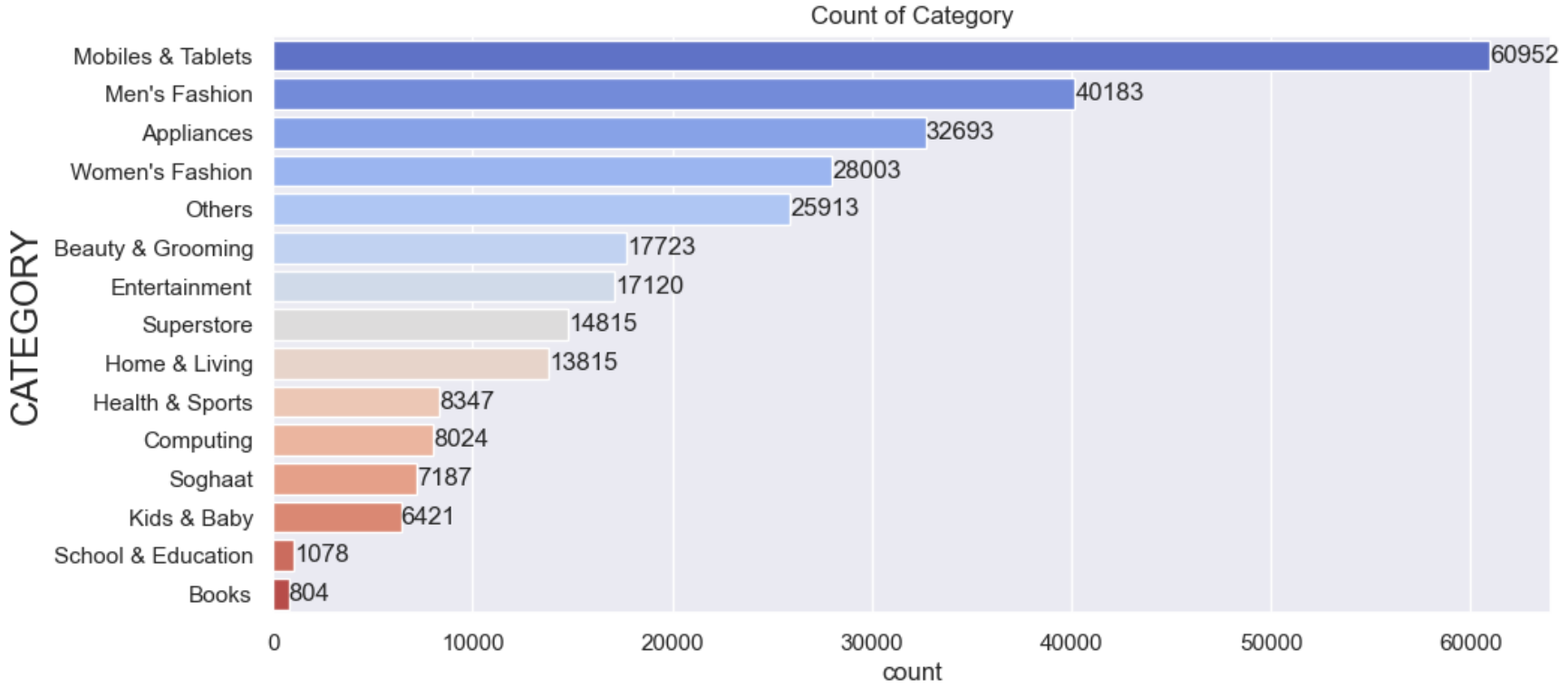
Exploratory Data Analysis

- We shall carry our
- Univariate,
- Bivariate and
- Multivariate Analysis on

Univariate Analysis

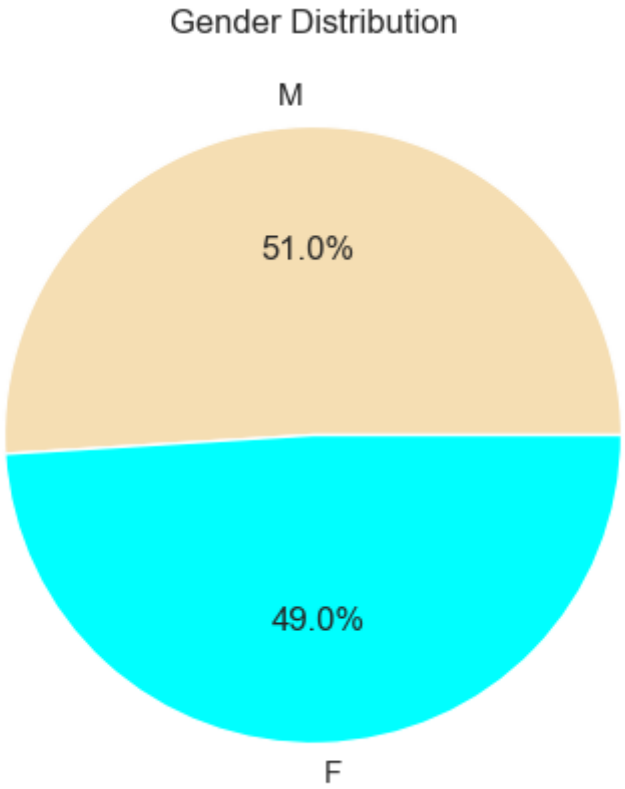
In [38]:

```
# COUNT OF CATEGORY

plt.figure(figsize=(11,5))
ax = sns.countplot(y=df["Category"], order = df["Category"].value_counts(ascending=False).index, palette='coolwarm
values = df["Category"].value_counts(ascending=False).values
ax.bar_label(container = ax.containers[0], labels=values)
plt.title("Count of Category")
plt.ylabel('CATEGORY', size='x-large');
```



Insight: The Mobile & Tablets, men'sfashion and Appliances are the top 3, while Books, School/education items are the least items ordered

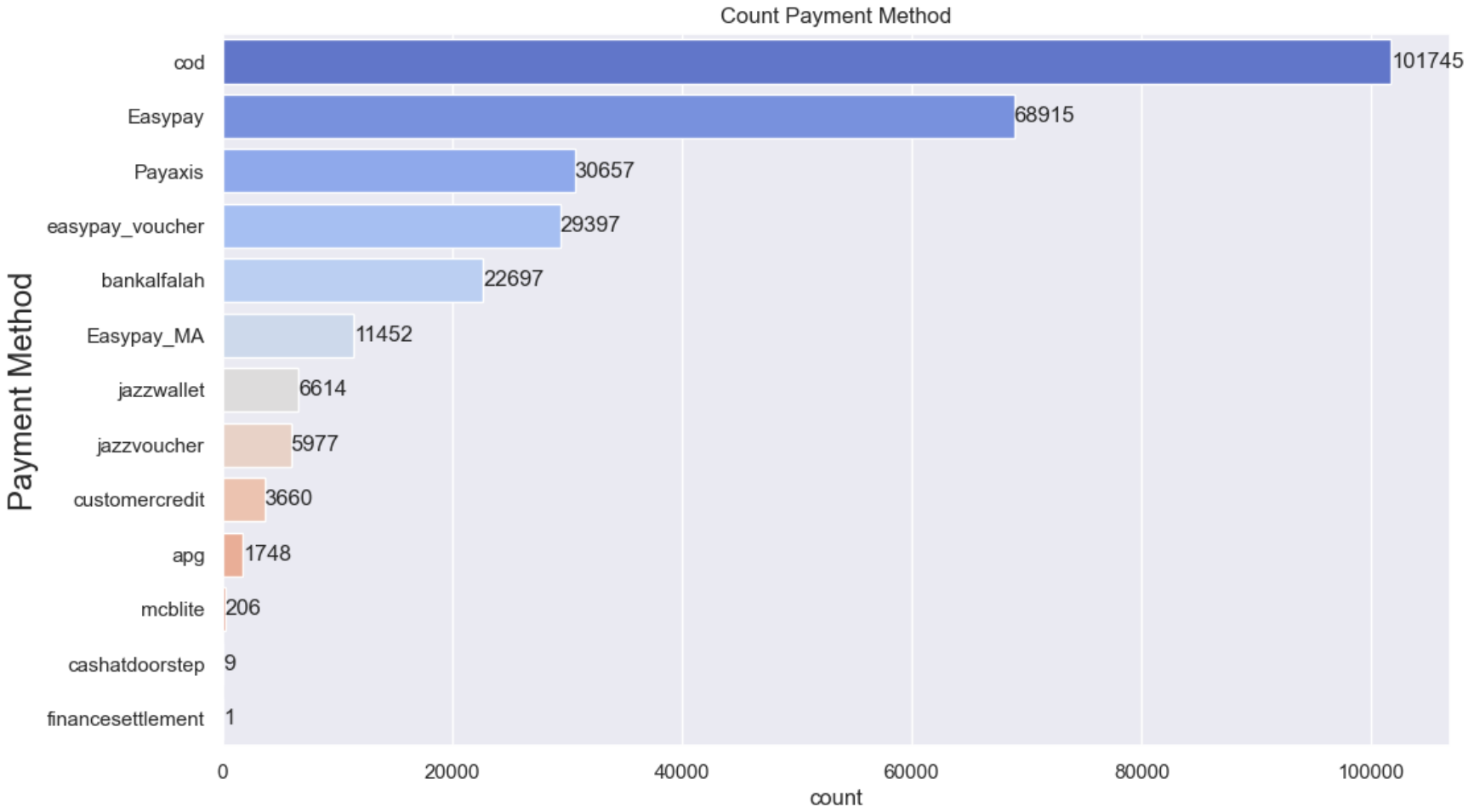
In [39]:
```
# Checking the Gender Distribution
Gender_counts = df['Gender'].value_counts()
plt.figure(figsize=(7,5))
plt.pie(Gender_counts, labels=Gender_counts.index, autopct='%1.1f%%', colors= ['wheat', 'cyan'])
plt.title('Gender Distribution');
```



Insight: We have an almost equal distribution in the Gender, with male slightly higher than female.

In [40]:

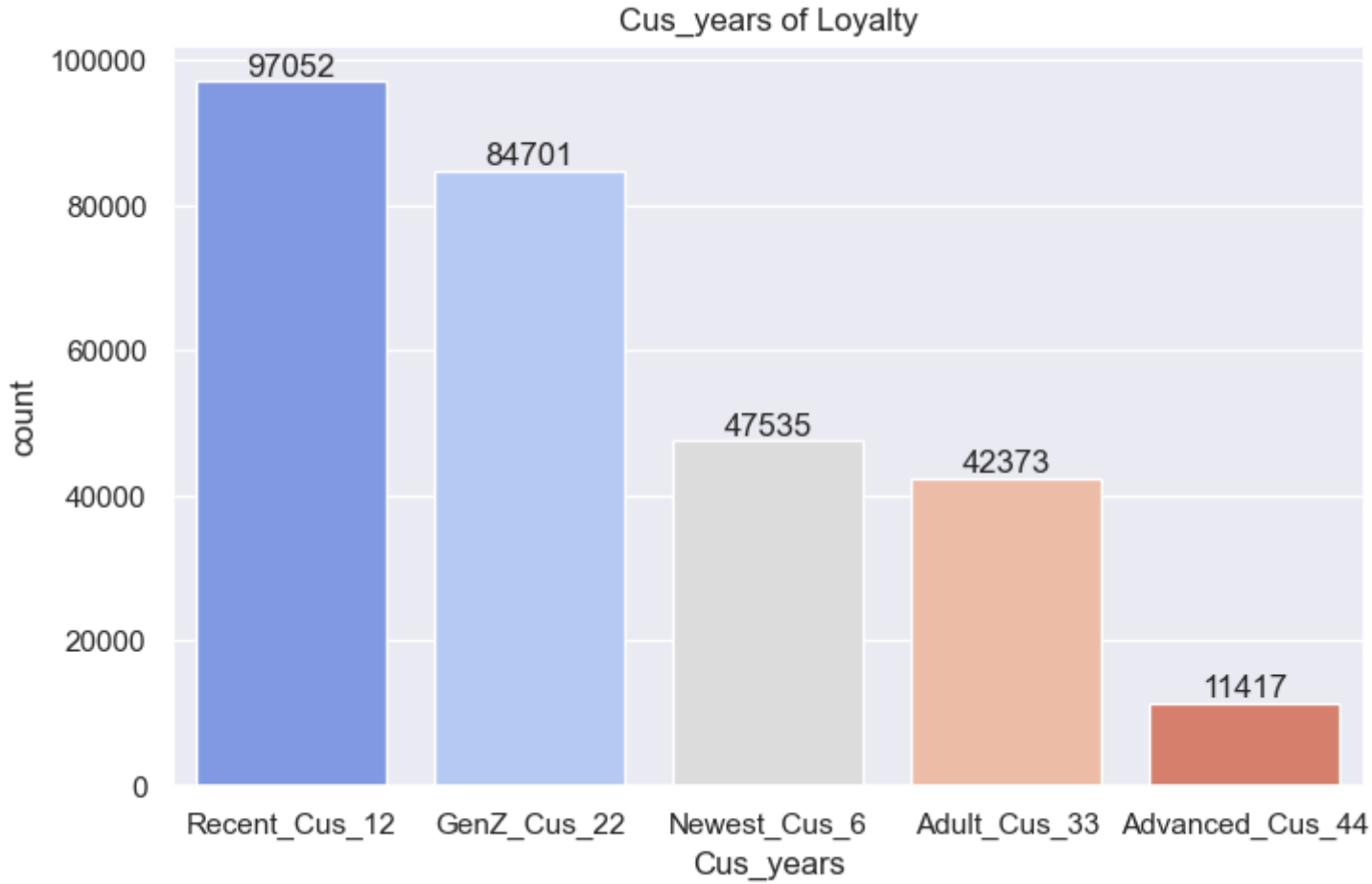
```
# Count of Payment methods
plt.figure(figsize=(12,7))
sns.set(font_scale=1.0)
ax = sns.countplot(y=df["PaymentMethod"], order = df["PaymentMethod"].value_counts(ascending=False).index, palette
values = df["PaymentMethod"].value_counts(ascending=False).values
ax.bar_label(container = ax.containers[0], labels=values)
plt.title("Count Payment Method")
plt.ylabel('Payment Method', size='x-large');
```


Count Payment Method
Payment Method
cod
Easypay
Payaxis
easypay_voucher
bankalfalah
Easypay_MA
jazzwallet
jazzvoucher
customercredit
apg
mcblite
cashatdoorstep
financesettlement
101745
68915
30657
29397
22697
11452
6614
5977
3660
1748
206
9
1
0
20000
40000
60000
80000
100000
count

Insight: Cod, easy pay are outstanding in our dataset and mostly used for customer payment, and financesettlement, cashatdoorstep, and mcblite are the least used

In [41]:

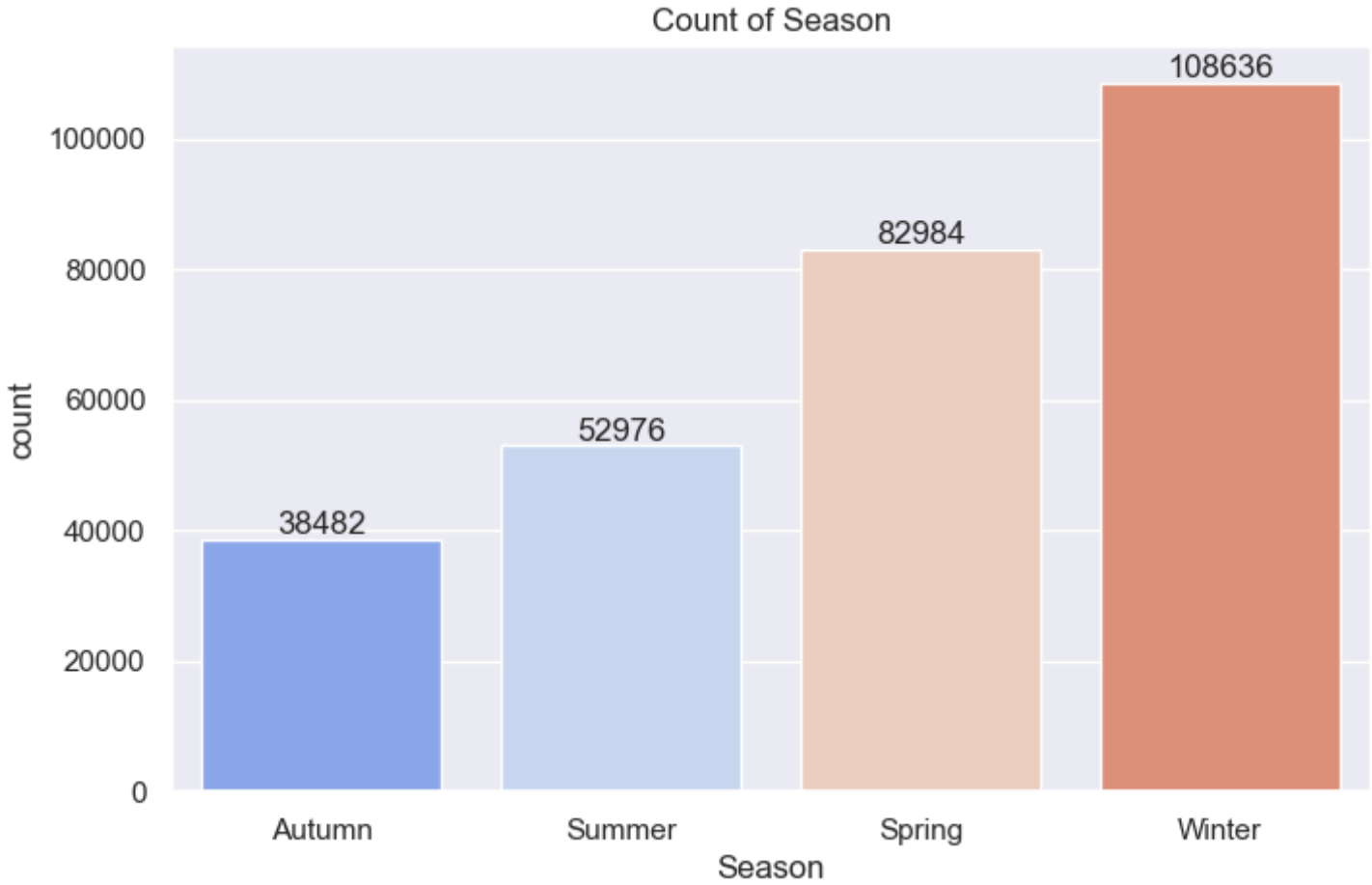
```
# The count of Customer Loyalty
plt.figure(figsize=(8,5))
ax = sns.countplot(x=df["Cus_years"], order = df["Cus_years"].value_counts(ascending=False).index, palette='coolwa
values = df["Cus_years"].value_counts(ascending=False).values
ax.bar_label(container = ax.containers[0], labels=values)
plt.title("Cus_years of Loyalty");
```



Insight: Recent (12yrs) and GenZ (22) customers are more in numbers, with advanced customers(44yrs) as the least

In [42]:

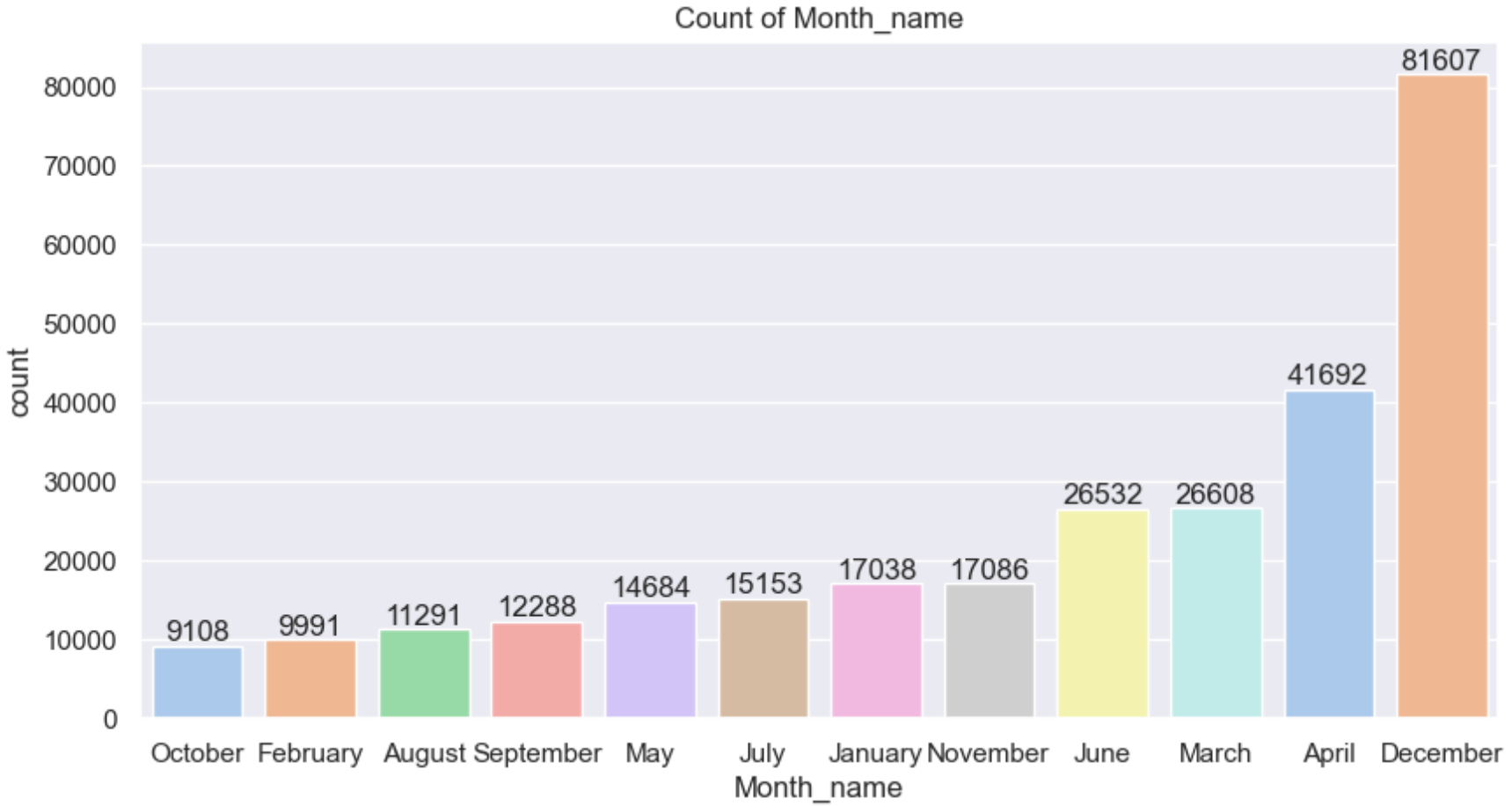
```
# We shall see the season mostly prominent for orders
plt.figure(figsize=(8,5))
ax = sns.countplot(x=df["Season"], order = df["Season"].value_counts(ascending=True).index, palette='coolwarm')
values = df["Season"].value_counts(ascending=True).values
ax.bar_label(container = ax.containers[0], labels=values)
plt.title("Count of Season");
```



Insight: Winter and Spring are most prominent and possibly the busiest for sterling E-commerce while Summer and Autum are light

In [43]:

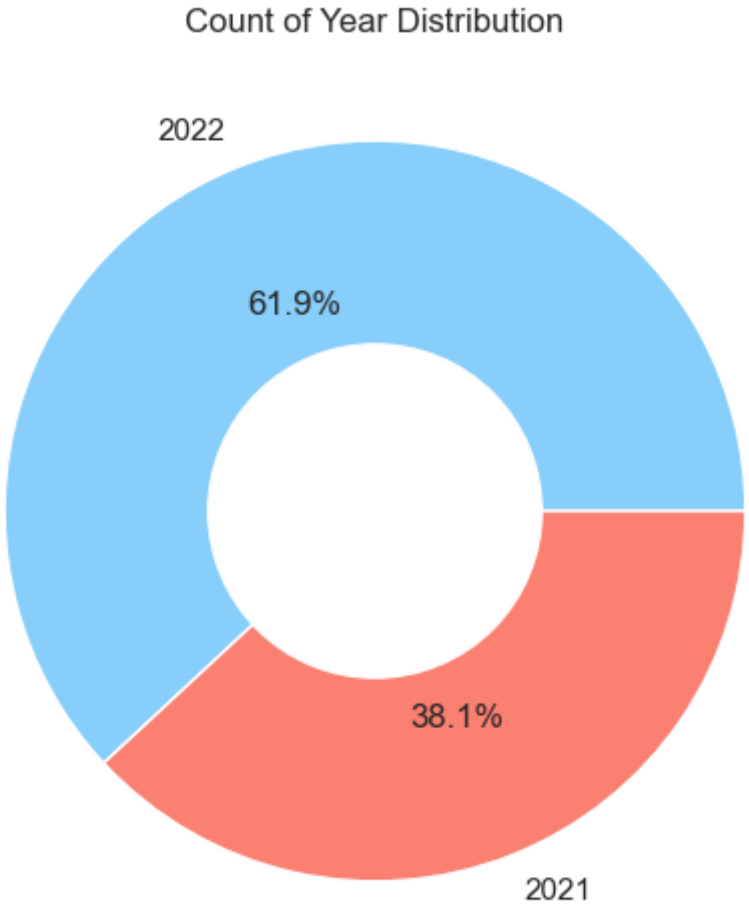
```
# We shall also check the months where we have most orders
plt.figure(figsize=(10,5))
ax = sns.countplot(x=df["Month_name"], order = df["Month_name"].value_counts(ascending=True).index, palette='paste
values = df["Month_name"].value_counts(ascending=True).values
ax.bar_label(container = ax.containers[0], labels=values)
plt.title("Count of Month_name");
```



Insight: December is Winter, while march and April are spring in most seasons and this is the period Sterling E-commerce experience most of their orders. October, Febraury are the most quiet periods.

In [44]:

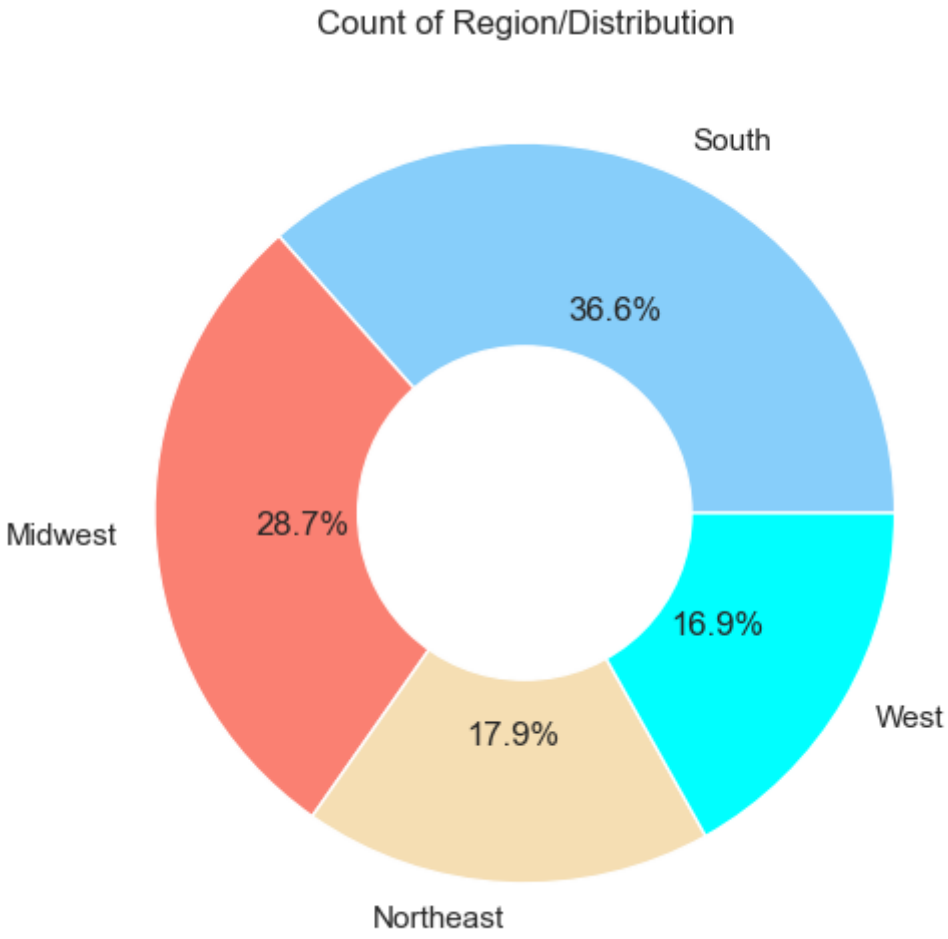
```
# We shall check which Year are most orders coming from
Year_counts = df['Year'].value_counts()
plt.figure(figsize=(8,6))
plt.pie(Year_counts, labels=Year_counts.index, autopct='%1.1f%%', colors= ['lightskyblue', 'salmon', 'wheat', 'cya

# draw circle
centre_circle = plt.Circle((0, 0), 0.45, fc='white')
fig = plt.gcf()
# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)
# Adding Title of chart
plt.title('Count of Year Distribution');
# Displaying Chart
plt.show()
```


Count of Year Distribution
2022
61.9%
38.1%
2021

Insight: Year 2022 is more than 2021 in our dataset

In [45]:

```
# We shall check which Region are most orders coming from
Region_counts = df['Region'].value_counts()
plt.figure(figsize=(8,6))
plt.pie(Region_counts, labels=Region_counts.index, autopct='%1.1f%%', colors= ['lightskyblue', 'salmon', 'wheat',

# draw circle
centre_circle = plt.Circle((0, 0), 0.45, fc='white')
fig = plt.gcf()
# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)
# Adding Title of chart
plt.title('Count of Region/Distribution');
# Displaying Chart
plt.show()
```


Count of Region/Distribution
South
36.6%
Midwest
28.7%
16.9%
West
17.9%
Northeast

Insight:South region followed by Midwest region have the most number of customers , with the least being the west.

In [46]:
```
df.columns
```

Out[46]:
```
Index(['Category', 'City', 'County', 'CustId', 'CustomerSince', 'Date',
       'FullName', 'Gender', 'ItemId', 'OrderId', 'PaymentMethod', 'PlaceName',
       'RefNum', 'Region', 'State', 'UserName', 'Zip', 'QtyOrdered', 'Total',
       'Year_1st_order', 'Year', 'Month', 'Month_name', 'Day_name', 'Quarter',
       'Cus_years', 'Season', 'Cus_years_Loyalty'],
      dtype='object')
```

Bivariate Analysis

- Emphasis on the Total – Total amount paid by customer

In [47]:
```
# We shall calculate the Total amount paid by Customer by product category

prod_cat = df.groupby('Category')['Total'].sum().astype(int).sort_values(ascending=False)
prod_cat
```

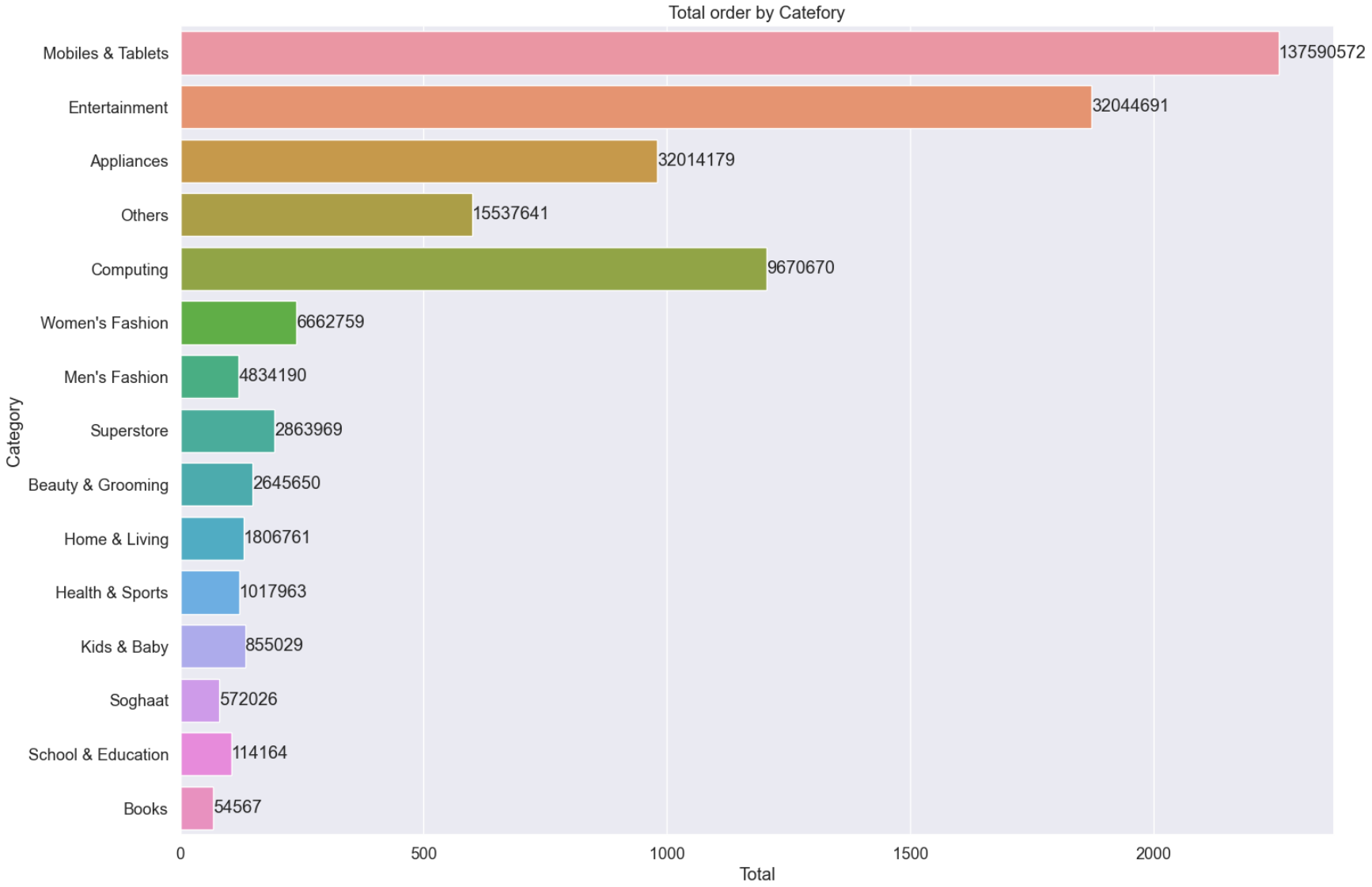
Out[47]:
```
Category
Mobiles & Tablets     137590572
Entertainment          32044691
Appliances             32014179
Others                 15537641
Computing               9670670
Women's Fashion         6662759
Men's Fashion           4834190
Superstore              2863969
Beauty & Grooming       2645650
Home & Living           1806761
Health & Sports         1017963
Kids & Baby              855029
Soghaat                  572026
School & Education       114164
Books                     54567
Name: Total, dtype: int32
```

In [49]:

```
# We shall visualise the Total Sums from each category as below


plt.figure(figsize=(17,12))
sns.set(font_scale=1.2)
ax = sns.barplot(y=df['Category'], x=df['Total'], ci=None, data=df, order=prod_cat.index)
values = prod_cat.values
ax.bar_label(container=ax.containers[0], labels=values);

plt.title('Total order by Catefory')

# Add labels to each bar
#for index, row in prod_cat.iterrows():
    #ax.text(index, row.Total + 1, round(row.Total, 2), color='black', ha="center")
```

Out[49]:

```
Text(0.5, 1.0, 'Total order by Catefory')
```


Total order by Catefory
Mobiles & Tablets
Entertainment
Appliances
Others
Computing
Women's Fashion
Men's Fashion
Superstore
Beauty & Grooming
Home & Living
Health & Sports
Kids & Baby
Soghaat
School & Education
Books
137590572
32044691
32014179
15537641
9670670
6662759
4834190
2863969
2645650
1806761
1017963
855029
572026
114164
54567
Category
0
500
1000
1500
2000
Total

Insight: Mobile/Tablets, Entertainment, Appliances and computing are mostly the products ordered

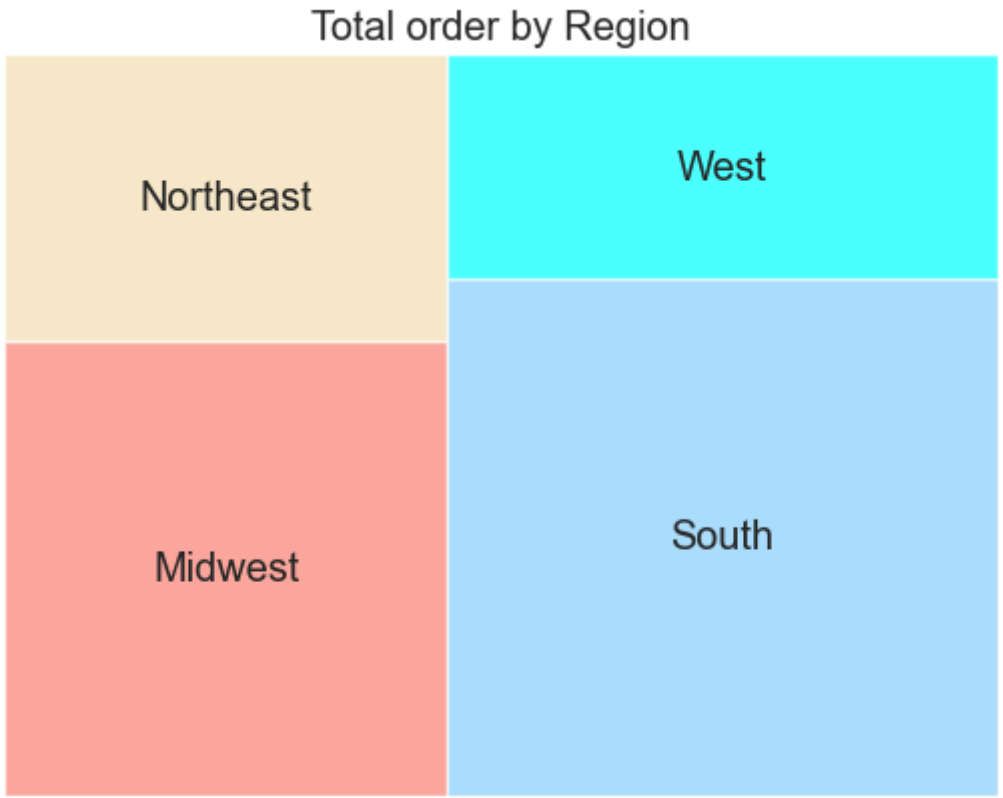
In [50]:
```
# We shall see the region ordering more
Reg1 = df.groupby('Region')[['Total']].sum().astype(int)
Reg1
```

Out[50]:

Region	Total
Midwest	67997231
Northeast	42810623
South	95947912
West	41529069

In [51]:

```
# Visualise using a TreeMap
squarify.plot(
    sizes=Reg1.values,
    #color=['Violet','Salmon','Khaki', 'Cyan'],
    color=['salmon','wheat','lightskyblue','cyan'],
    label=Reg1.index,
    alpha=0.7,
)
plt.title('Total order by Region')
plt.axis('off');
```



Insight: The total order mostly are coming also from the south and then midwest, the Northeast and west remains the least

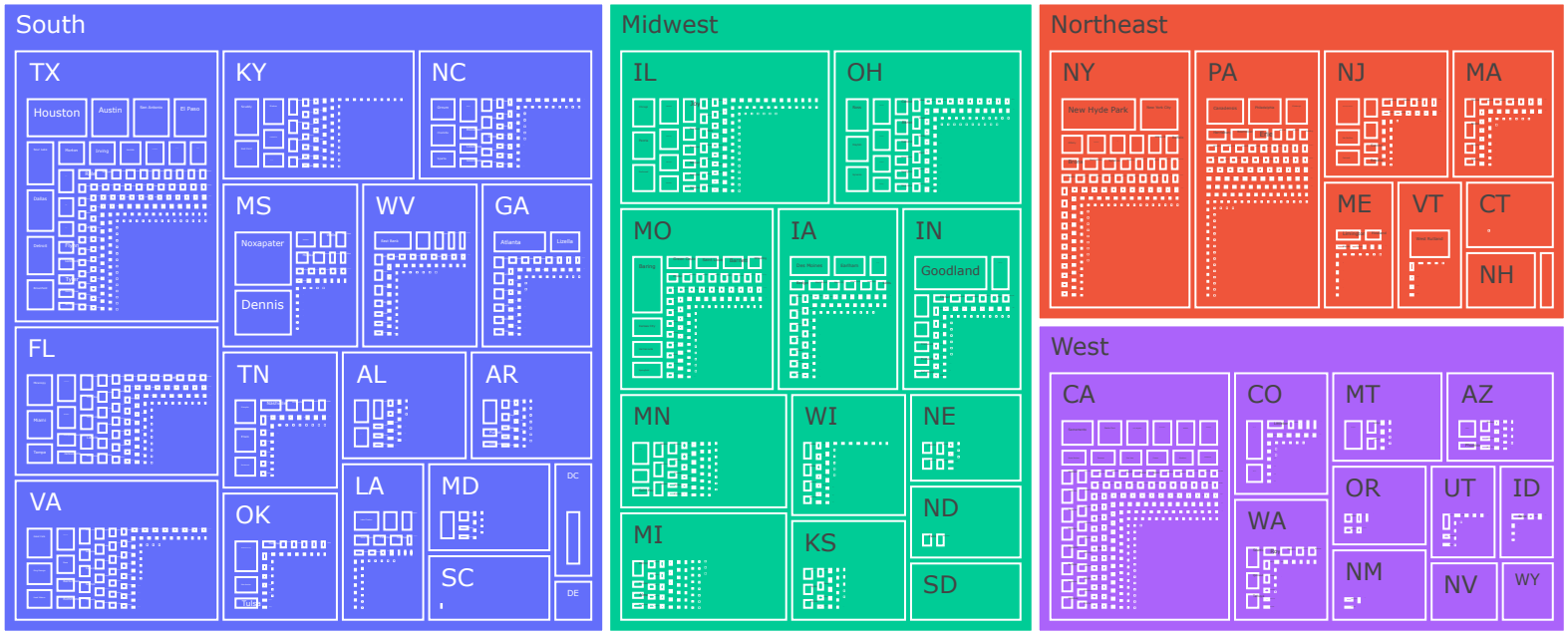
In [52]:

```
import plotly.express as px
import plotly
```

In [53]:

```
# We shall Visualise the Total amount paid by Customers by Region, State, and City


state = df['State']
region = df['Region']
city = df['City']
county = df['County']
total = df['Total']
quantity = ['QtyOrdered']
fig =px.treemap(df,
                path=[region, state, city],
                values=total,
                color=region,
                color_continuous_scale=['red','yellow','green'],
                title='Overview of Total amount paid by Customer by Region, State & City')


fig.show()
```

Overview of Total amount paid by Customer by Region, State & City




```
In [62]: df['Total'].sum().astype(int)
```

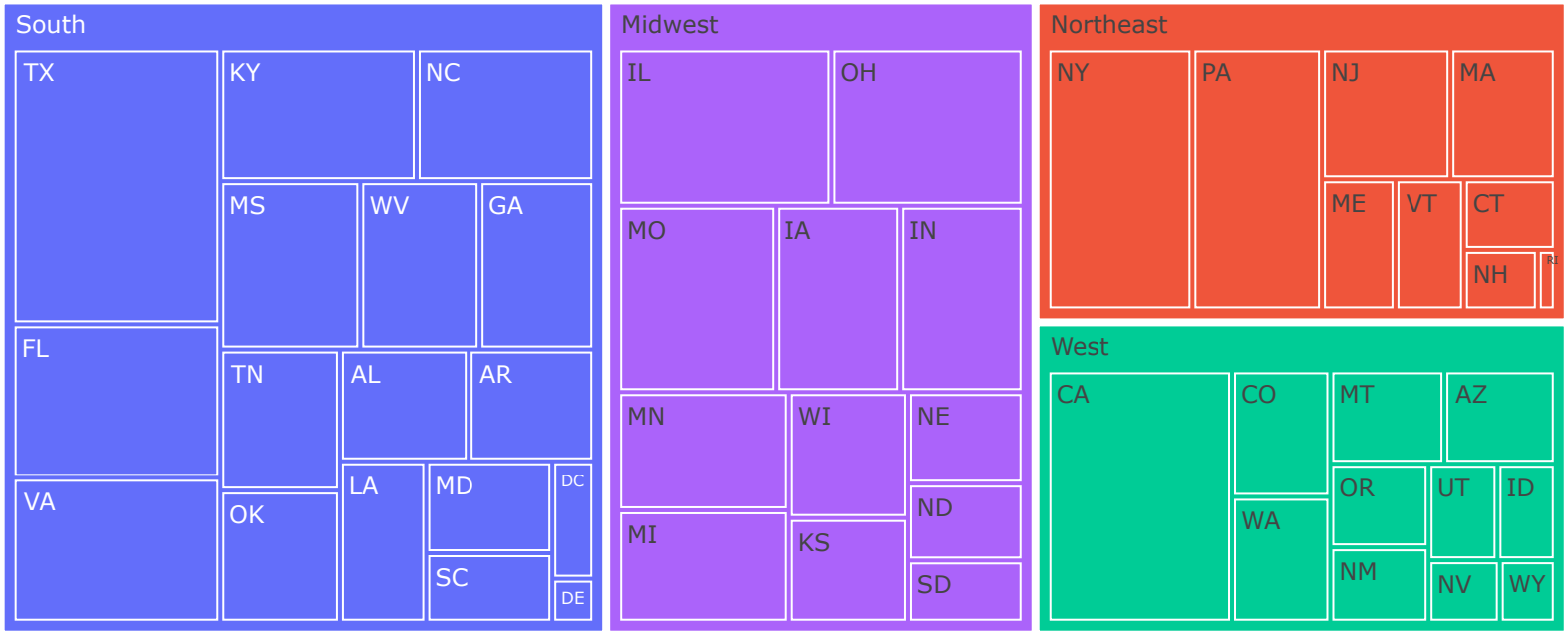
```
Out[62]: 248284838
```

Insight: We have broken down the Total sales by State and we can see the regions, distribution of the states and city using the interactive map

In [68]:

```
# We shall Visualise the Total amount paid by Customers by Region & Staate


state = df['State']
region = df['Region']
city = df['City']
county = df['County']
total = df['Total']

fig =px.treemap(df,
                path=[region, state,],
                values=total,
                color=region,
                color_continuous_scale=['red','yellow','green','yellow'],
                title='Overview of Total amount paid by Customer by Region & State')
fig.show()
```

Overview of Total amount paid by Customer by Region & State



Insight: Further seeing the details in the total order, we can see the cities with higest/lowest order by state in the various regions. South have texas and florida, west have Carlifornia as the highest sales

In [73]:

```
# We shall Visualise the Total amount paid by Customers by State, and City


state = df['State']
region = df['Region']
city = df['City']
county = df['County']
total = df['Total']


fig =px.treemap(df,
                path=[county],
                values=total,
                color=county,
                color_continuous_scale=['red','yellow','green','yellow'],
                title='Overview of Total amount paid by Customer by County')
fig.show()
```

Overview of Total amount paid by Customer by County

Insight: L.A. County returns the highest Total of order, but we shall visualise and present only the top 10 counties below.

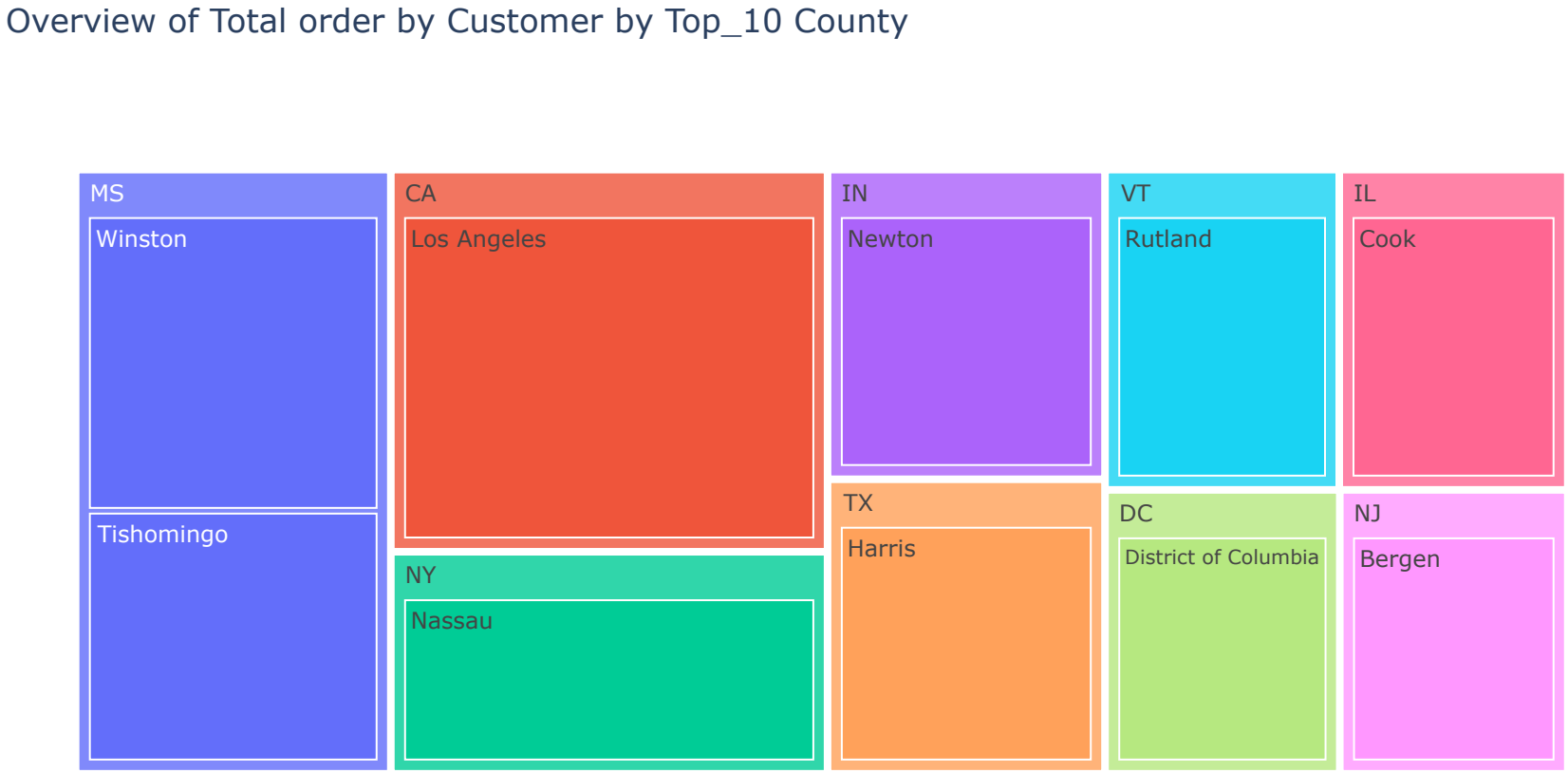
In [183]:
```
# Check the top Top 10 States, County
xxx = df.groupby(['State','County'])['Total'].sum().sort_values(ascending=False).head(10).reset_index(name='Total_
xxx
```

Out[183]:

	State	County	Total_order
0	CA	Los Angeles	3.132349e+06
1	MS	Winston	1.930524e+06
2	NY	Nassau	1.819846e+06
3	MS	Tishomingo	1.643737e+06
4	IN	Newton	1.612524e+06
5	TX	Harris	1.533607e+06
6	VT	Rutland	1.407923e+06
7	IL	Cook	1.373597e+06
8	DC	District of Columbia	1.250829e+06
9	NJ	Bergen	1.214836e+06

In [181]:

```
px.treemap(data_frame=xxx, path=['State','County'], values='Total_order',\
           title='Overview of Total order by Customer by Top_10 County')
```



Insight: L.A. County returns the highest Total of orders and by research is the most populous county in the United States. Also Jefferson and Monroe are doing good.

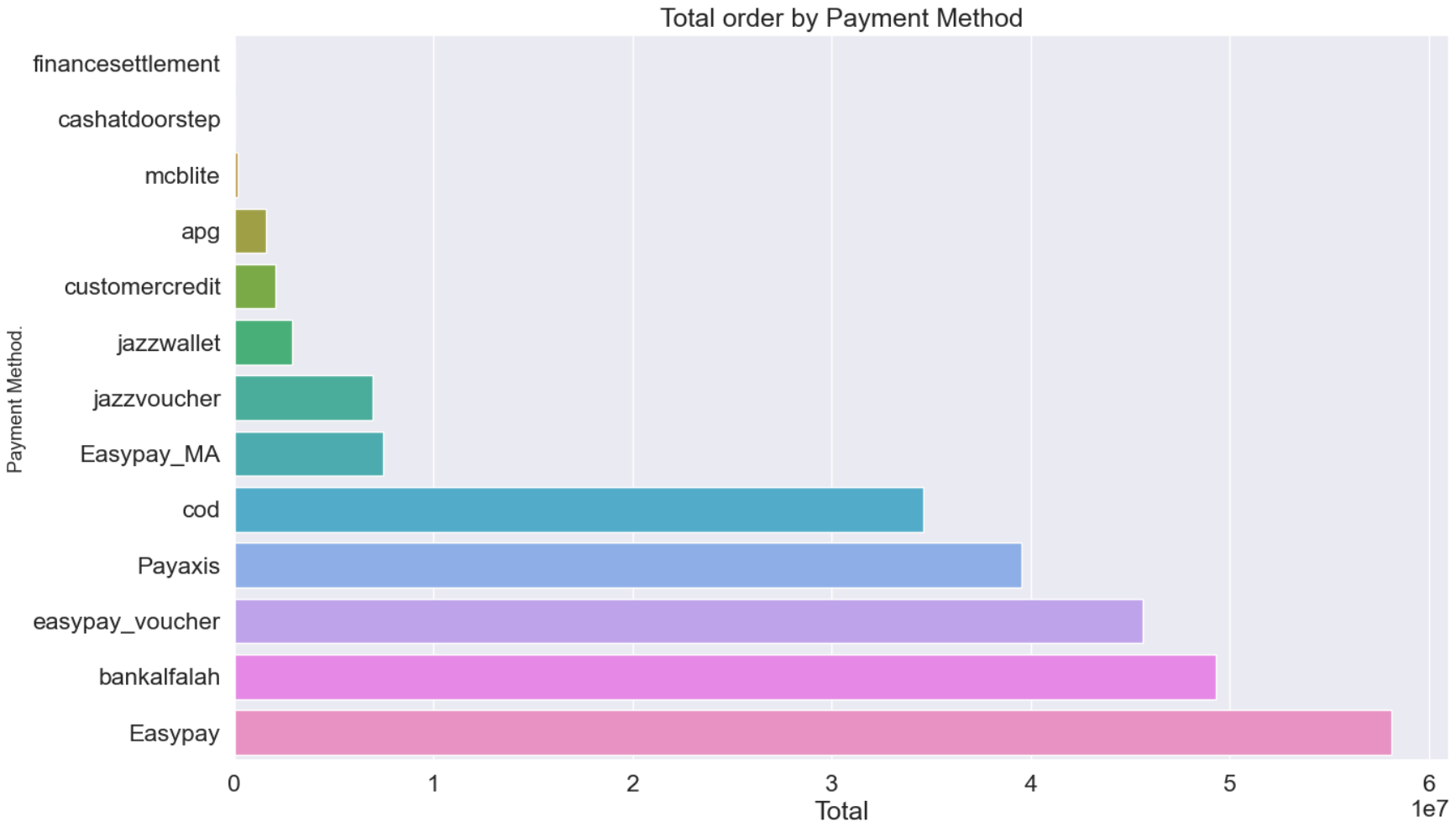
In [100]:
```
# We shall group the payment methods and sum the total
pay = df.groupby('PaymentMethod')[['Total']].sum().sort_values(by='Total').astype(int).reset_index()
pay
```

Out[100]:

	PaymentMethod	Total
0	financesettlement	2059
1	cashatdoorstep	3799
2	mcblite	163340
3	apg	1599187
4	customercredit	2037836
5	jazzwallet	2912487
6	jazzvoucher	6927541
7	Easypay_MA	7463977
8	cod	34618106
9	Payaxis	39539225
10	easypay_voucher	45601649
11	bankalfalah	49302494
12	Easypay	58113130

In [101]:

```
# Visualise the above
plt.figure(figsize=(15,9))
sns.set(font_scale=1.5)
plt.title('Total order by Payment Method')
sns.barplot(x='Total', y='PaymentMethod', data=pay)
plt.ylabel('Payment Method.', fontsize=13);
```



Insight: Easypay Bankafalah and easypay_voucher are the most payment methods used against the Total orders.

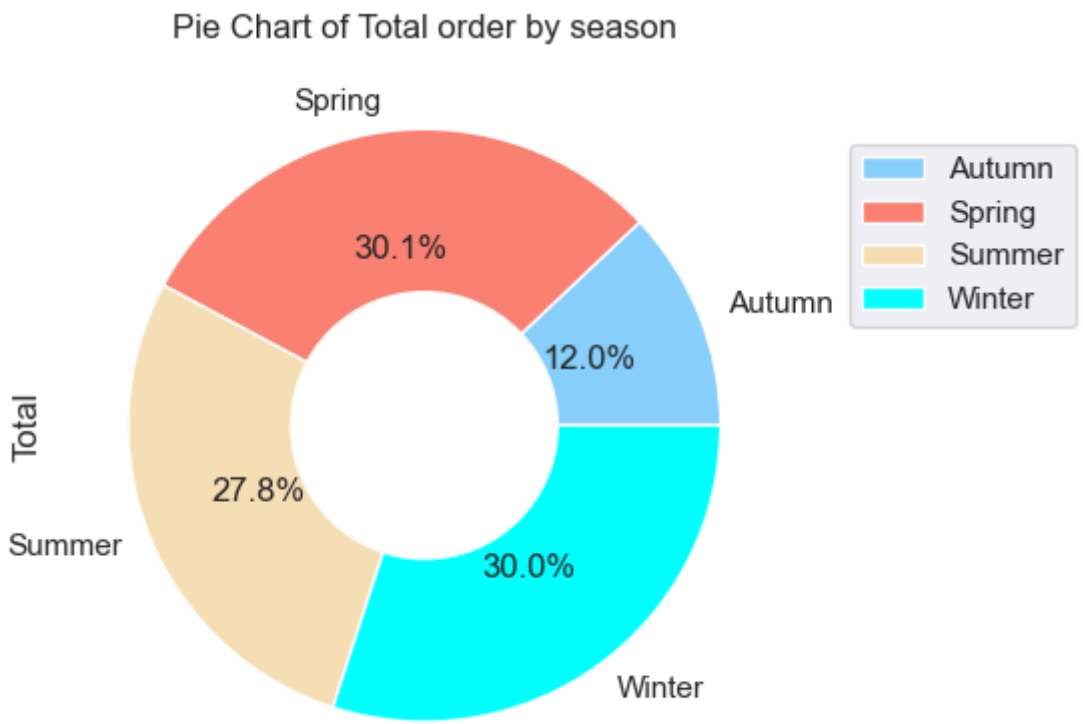
In [102]:
```
# We shall group the season and sum the total
season = df.groupby('Season')[['Total']].sum().sort_values(by='Total').astype(int).reset_index()
season
```

Out[102]:

	Season	Total
0	Autumn	29894710
1	Summer	69125230
2	Winter	74484675
3	Spring	74780221

In [104]:

```
# plot pie chart
sns.set(font_scale=1.0)
ax = df.groupby(['Season']).sum().plot(kind='pie', y='Total',autopct='%1.1f%%', colors=['lightskyblue', 'salmon',
ax.legend(loc='upper right',bbox_to_anchor=(1.4, 0.9))
# draw circle
centre_circle = plt.Circle((0, 0), 0.45, fc='white')
fig = plt.gcf()
# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)
plt.show()
```



Insight: Most of the orders are made in Spring and Winter

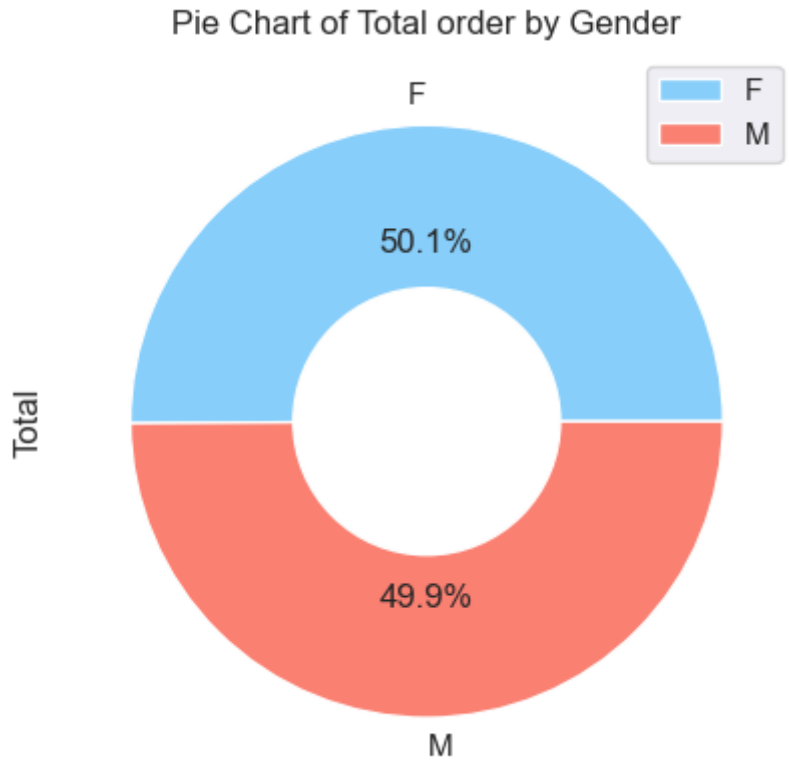
In [105]:
```
# Which Gender have the highest Total order
gender = df.groupby('Gender')[['Total']].sum().sort_values(by='Total').astype(int).reset_index()
gender
```

Out[105]:

	Gender	Total
0	M	123895666
1	F	124389171

In [106]:

```
# plot pie chart
df.groupby(['Gender']).sum().plot(kind='pie', y='Total',autopct='%1.1f%%', colors=['lightskyblue', 'salmon'], titl

# draw circle
centre_circle = plt.Circle((0, 0), 0.45, fc='white')
fig = plt.gcf()
# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)
plt.show()
```



Insight: The Female Gender have the most order, although the difference is infinitesimal

In [107]:
```
cus_loyalty = df.groupby('Cus_years')[['Total']].sum().sort_values(by='Total').astype(int).reset_index()
cus_loyalty
```

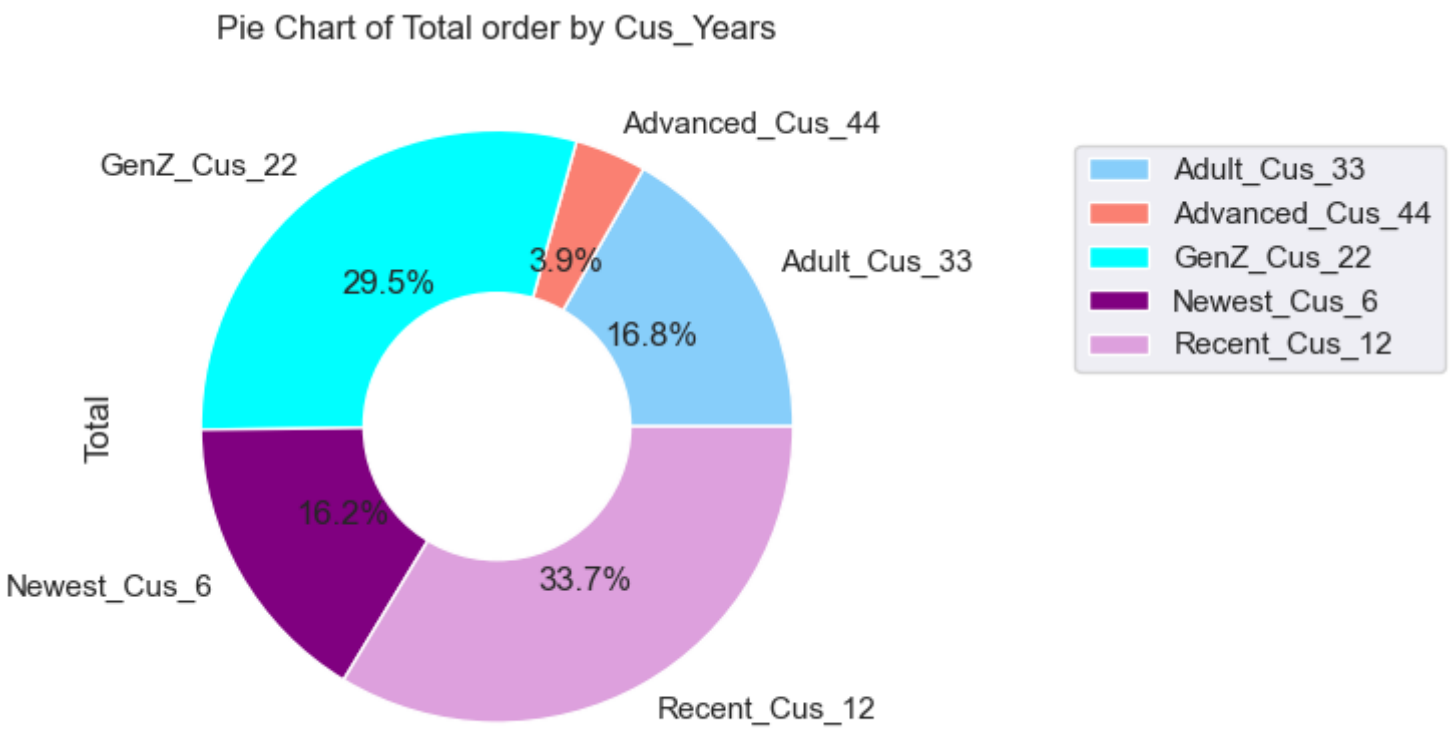
Out[107]:

	Cus_years	Total
0	Advanced_Cus_44	9674793
1	Newest_Cus_6	40113709
2	Adult_Cus_33	41644051
3	GenZ_Cus_22	73258868
4	Recent_Cus_12	83593415

In [108]:

```
# plot pie chart
plt.figure(figsize=(10,7))
ax = df.groupby(['Cus_years']).sum().plot(kind='pie', y='Total',autopct='%1.1f%%',  labeldistance=1.1, colors=['li
ax.legend(loc='upper right',bbox_to_anchor=(1.8, 0.9))
# draw circle
centre_circle = plt.Circle((0, 0), 0.45, fc='white')
fig = plt.gcf()
# Adding Circle in Pie chart
fig.gca().add_artist(centre_circle)
plt.show();
```

```
<Figure size 1000x700 with 0 Axes>
```



Insight: The recent_cus(12yrs) are most followed by Genz_customers (22yrs)

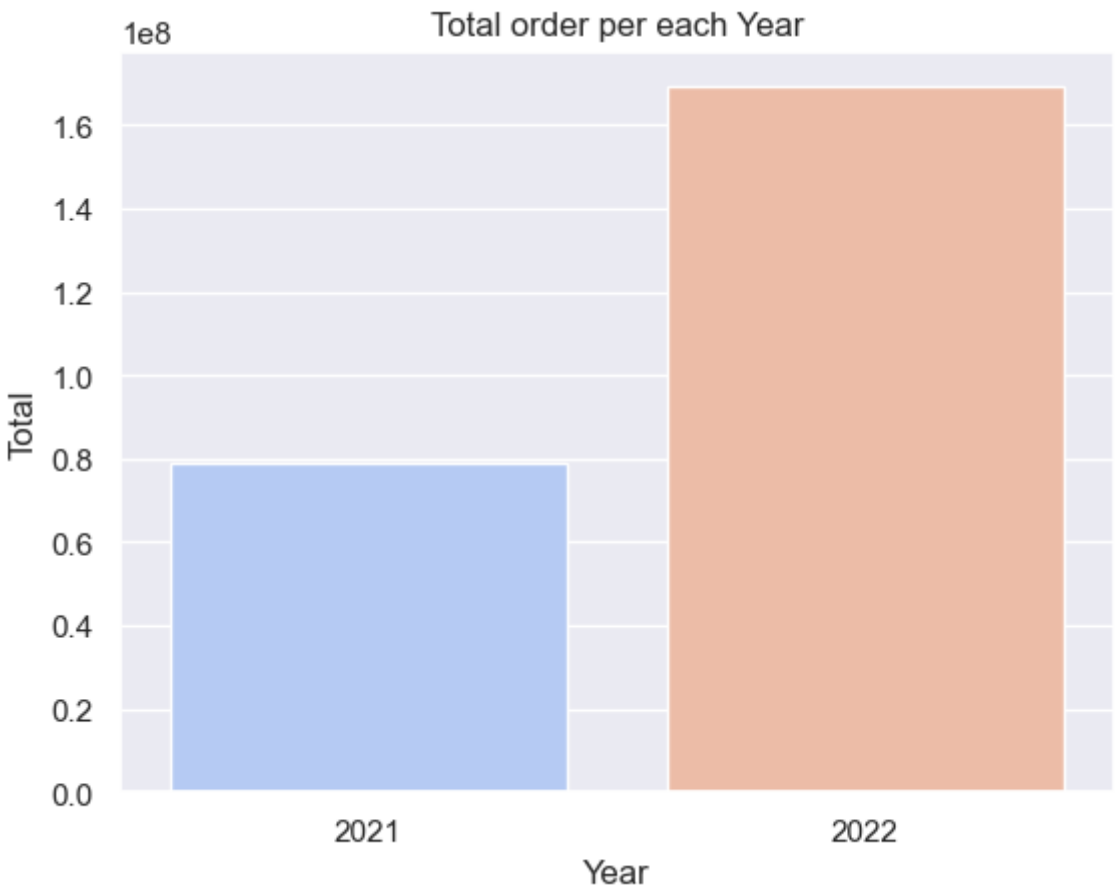
In [109]:
```
# Plot of Total Amount by years
yr = df.groupby('Year')['Total'].sum().reset_index()
yr
```

Out[109]:

	Year	Total
0	2021	7.903669e+07
1	2022	1.692482e+08

In [110]:

```
plt.title('Total order per each Year')
sns.barplot(y='Total', x='Year', data=yr, palette='coolwarm');
```



Insight: We have more order in 2022 than the previous year of 2021

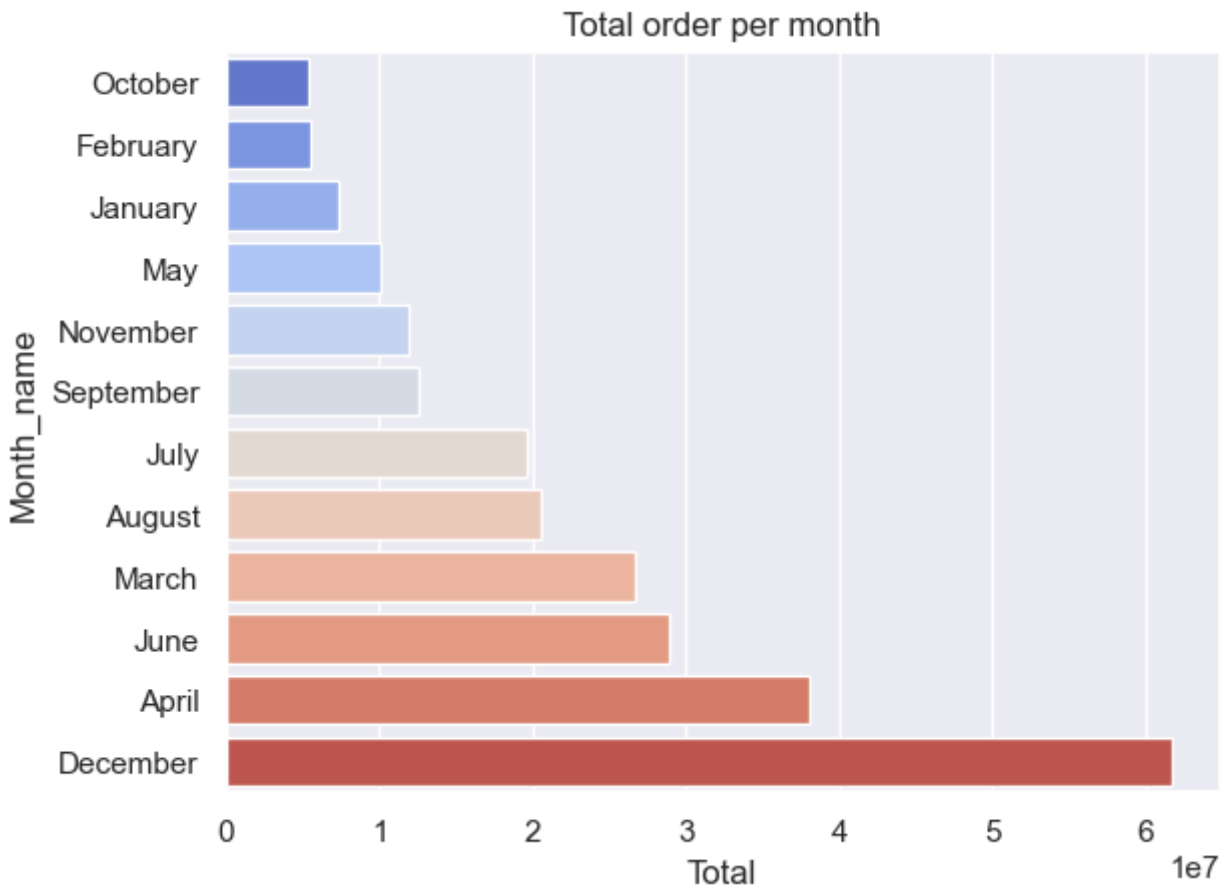
In [111]:
```
# Plot of Total Amount by Month_name
month_n = df.groupby('Month_name')['Total'].sum().reset_index().sort_values(by='Total')
month_n
```

Out[111]:

	Month_name	Total
10	October	5.443066e+06
3	February	5.466883e+06
4	January	7.305089e+06
8	May	1.009866e+07
9	November	1.188092e+07
11	September	1.257073e+07
5	July	1.964355e+07
1	August	2.059771e+07
7	March	2.664865e+07
6	June	2.888398e+07
0	April	3.803291e+07
2	December	6.171270e+07

In [112]:

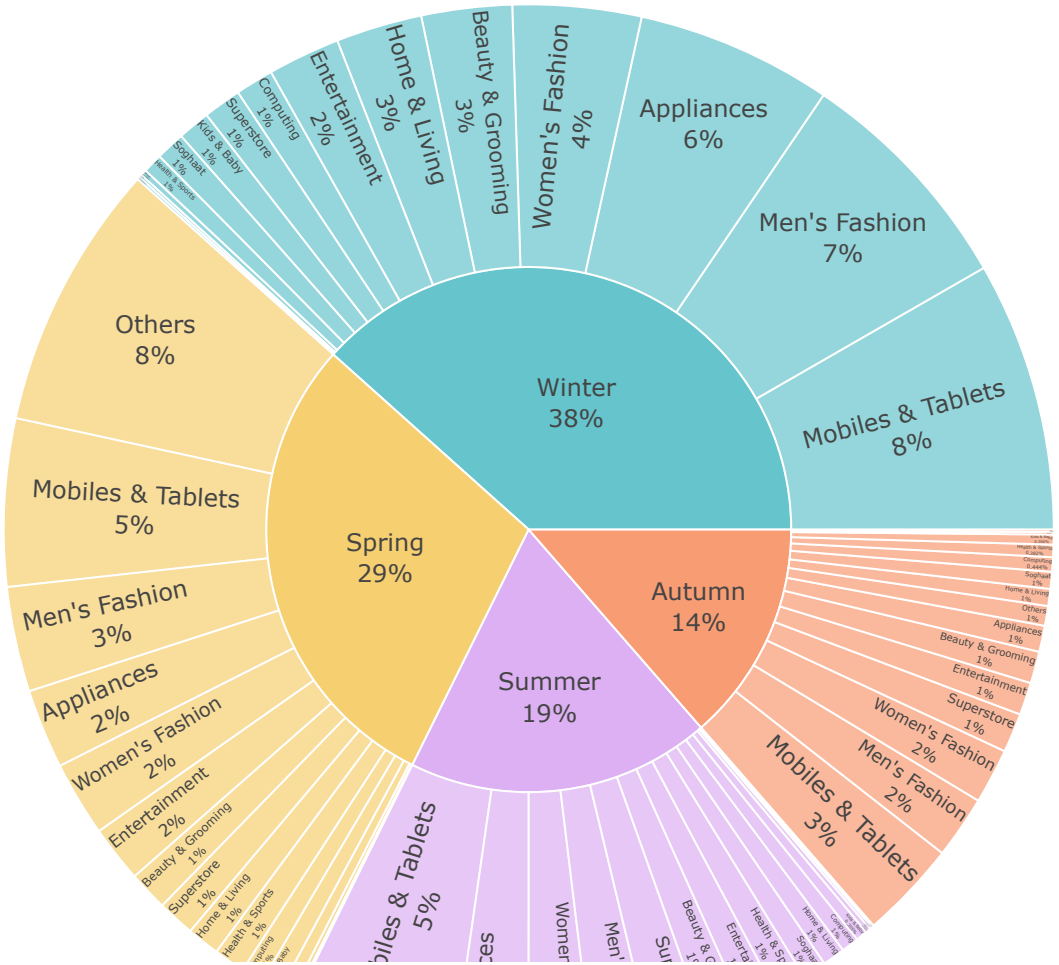
```
plt.title('Total order per month')
sns.barplot(x='Total', y='Month_name', data=month_n, palette='coolwarm');
```



Insight: December, April and June have the highest Total orders, while October, January and February are the least

In [113]:

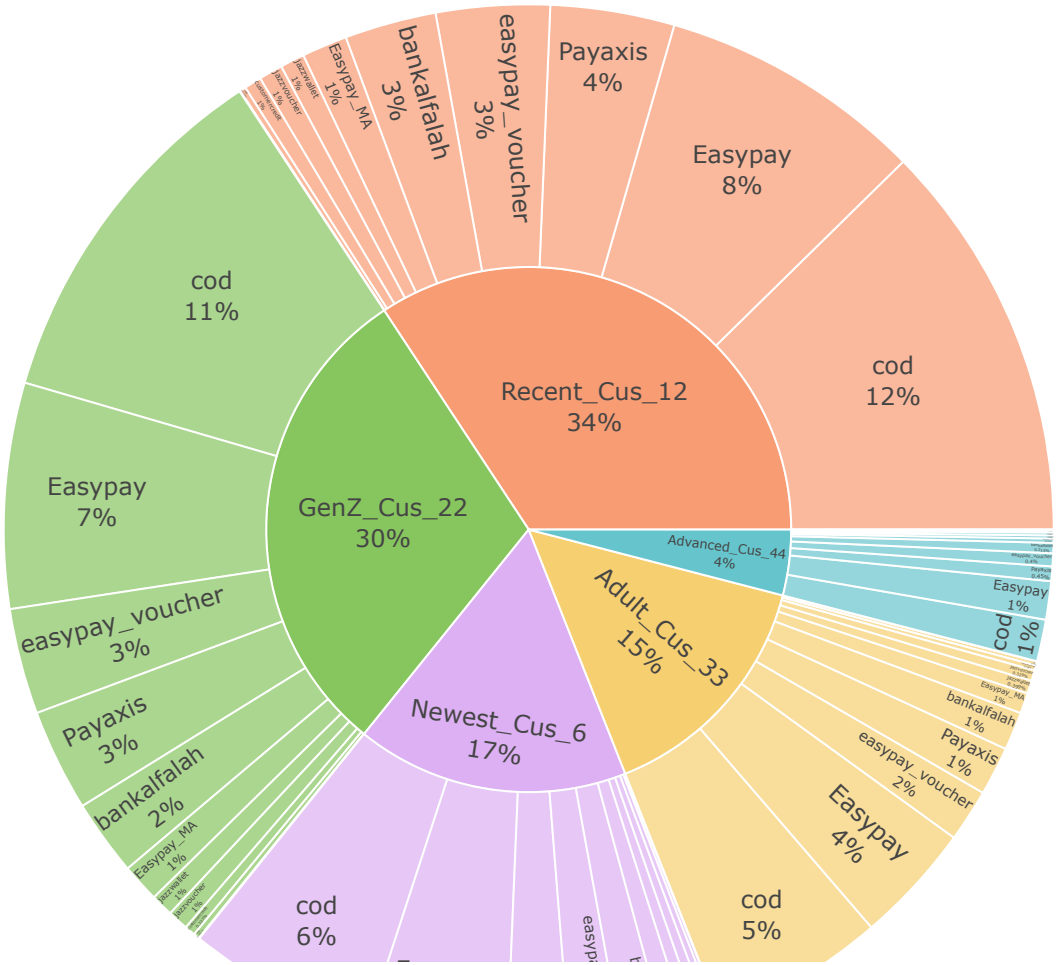
```
# We shall see what Customers are ordering more per season
fig = px.sunburst(
data_frame=df,
path=['Season', 'Category'],
color='Season',
color_discrete_sequence=px.colors.qualitative.Pastel,
maxdepth=-1)

fig.update_traces(textinfo='label+percent entry')
fig.update_layout(margin=dict(t=0, l=0, r=0, b=0))

fig.show()
```


Winter
38%
Spring
29%
Summer
19%
Autumn
14%
Mobiles & Tablets
8%
Men's Fashion
7%
Appliances
6%
Women's Fashion
4%
Beauty & Grooming
3%
Home & Living
3%
Entertainment
2%
Computing
1%
Superstore
1%
Kids & Baby
1%
Soghaat
1%
Health & Sports
1%
Others
8%
Mobiles & Tablets
5%
Men's Fashion
3%
Appliances
2%
Women's Fashion
2%
Entertainment
2%
Beauty & Grooming
1%
Superstore
1%
Home & Living
1%
Health & Sports
1%
Mobiles & Tablets
5%
Mobiles & Tablets
3%
Men's Fashion
2%
Women's Fashion
2%
Superstore
1%
Entertainment
1%
Beauty & Grooming
1%
Appliances
1%
Others
1%
Home & Living
1%
Soghaat
1%
Computing
1%
Health & Sports
1%

Insights: Most of the orders in the Winter, and Spring and still remains Mobiles & tablets, men and women fashions are also part of the top orders across seasons.

In [114]:

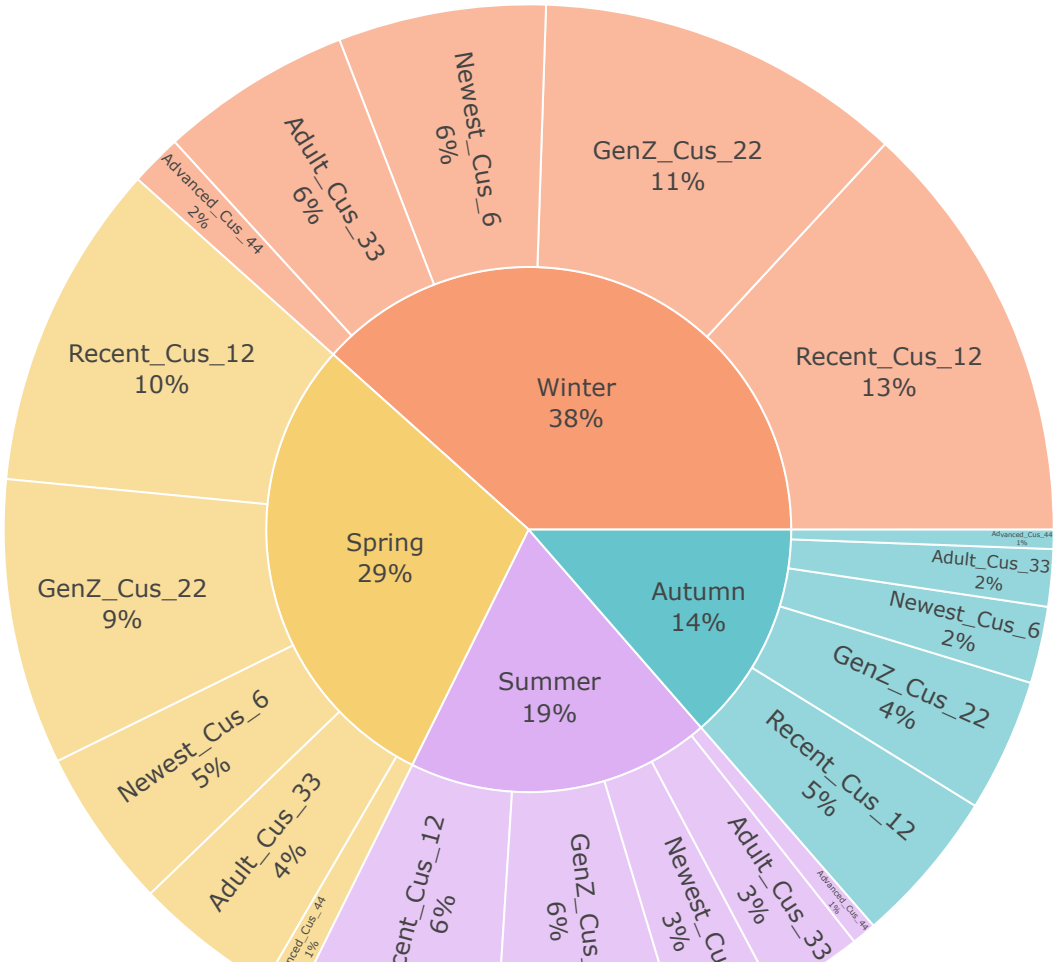
```
# We shall also see payment methods used by Different regions
fig = px.sunburst(
data_frame=df,
path=['Cus_years', 'PaymentMethod'],
color='Cus_years',
color_discrete_sequence=px.colors.qualitative.Pastel,
maxdepth=-1)

fig.update_traces(textinfo='label+percent entry')
fig.update_layout(margin=dict(t=0, l=0, r=0, b=0))

fig.show()
```


Recent_Cus_12
34%
cod
12%
Easypay
8%
Payaxis
4%
easypay_voucher
3%
bankalfalah
3%
Easypay_MA
1%
jazzwallet
1%
1link_voucher
1%
GenZ_Cus_22
30%
cod
11%
Easypay
7%
easypay_voucher
3%
Payaxis
3%
bankalfalah
2%
Easypay_MA
1%
jazzwallet
1%
1link_voucher
1%
Newest_Cus_6
17%
cod
6%
Adult_Cus_33
15%
cod
5%
Easypay
4%
easypay_voucher
2%
Payaxis
1%
bankalfalah
1%
Advanced_Cus_44
4%
cod
1%
Easypay
1%

Insight: Comparing the customer loyalty against payment methods, it would seems almost everyone prefers the COD and easypay across all ages.

In [115]:

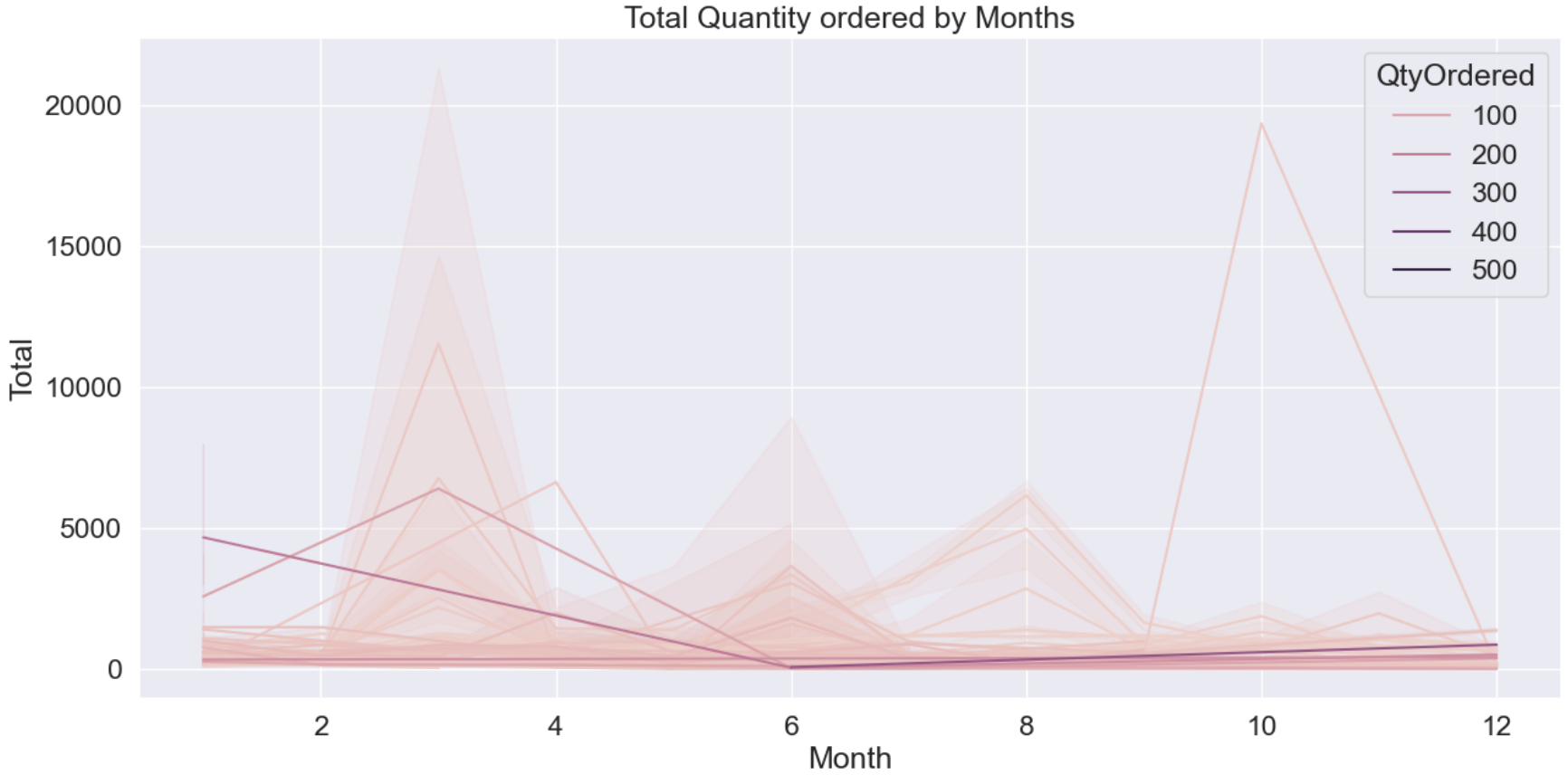
```
# Also comparing the Customer loyalty types by season

# We shall also see payment methods used by Different regions
fig = px.sunburst(
data_frame=df,
path=['Season', 'Cus_years'],
color='Season',
color_discrete_sequence=px.colors.qualitative.Pastel,
maxdepth=-1)

fig.update_traces(textinfo='label+percent entry')
fig.update_layout(margin=dict(t=0, l=0, r=0, b=0))

fig.show()
```


Winter
38%
Recent_Cus_12
13%
GenZ_Cus_22
11%
Newest_Cus_6
6%
Adult_Cus_33
6%
Advanced_Cus_44
2%
Spring
29%
Recent_Cus_12
10%
GenZ_Cus_22
9%
Newest_Cus_6
5%
Adult_Cus_33
4%
Summer
19%
Recent_Cus_12
6%
GenZ_Cus_22
6%
Newest_Cus_6
3%
Adult_Cus_33
3%
Autumn
14%
Recent_Cus_12
5%
GenZ_Cus_22
4%
Newest_Cus_6
2%
Adult_Cus_33
2%

Insight: No Particular distribution among our customer loyalty types in the various seasons

Multivariate Analysis

In [60]:
```
# Checking the Total per month per year
plt.figure(figsize=(15,7))
plt.title('Total Quantity ordered by Months')

sns.lineplot(data=df, x='Month', y='Total', hue='QtyOrdered', color='blue');
```



insight: Most of the total orders are maximum in December, march and april

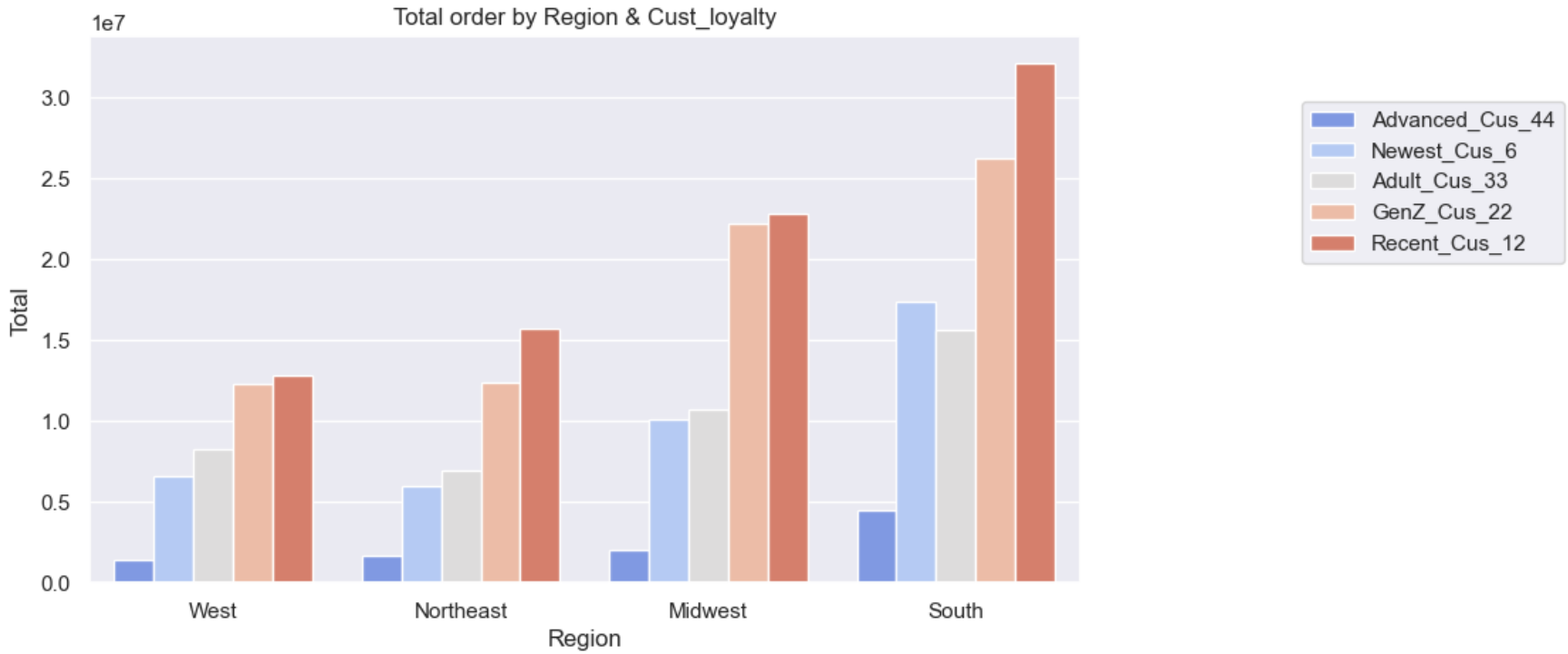
In [146]:
```
# Grouping the Region, Cus_years and Total
Cust = df.groupby(['Region', 'Cus_years'])['Total'].sum().reset_index().sort_values(by='Total')
Cust
```

Out[146]:

	Region	Cus_years	Total
16	West	Advanced_Cus_44	1.397800e+06
6	Northeast	Advanced_Cus_44	1.722447e+06
1	Midwest	Advanced_Cus_44	2.069086e+06
11	South	Advanced_Cus_44	4.485461e+06
8	Northeast	Newest_Cus_6	6.024519e+06
18	West	Newest_Cus_6	6.637813e+06
5	Northeast	Adult_Cus_33	6.969864e+06
15	West	Adult_Cus_33	8.295681e+06
3	Midwest	Newest_Cus_6	1.008705e+07
0	Midwest	Adult_Cus_33	1.075521e+07
17	West	GenZ_Cus_22	1.233891e+07
7	Northeast	GenZ_Cus_22	1.239103e+07
19	West	Recent_Cus_12	1.285887e+07
10	South	Adult_Cus_33	1.562329e+07
9	Northeast	Recent_Cus_12	1.570277e+07
13	South	Newest_Cus_6	1.736432e+07
2	Midwest	GenZ_Cus_22	2.223802e+07
4	Midwest	Recent_Cus_12	2.284785e+07
12	South	GenZ_Cus_22	2.629091e+07
14	South	Recent_Cus_12	3.218393e+07

In [147]:

```
# Check the Customer loyalty by Gender based on their total order.
plt.figure(figsize=(9,5))
plt.title('Total order by Region & Cust_loyalty')
ax = sns.barplot(x='Region',y='Total', data=Cust, hue='Cus_years', palette='coolwarm', ci=None)
ax.legend(loc='upper right',bbox_to_anchor=(1.5, 0.9));
```



Insight: The Customer loyalty is equally distributted by region as previously seen with most sales scoming from the South

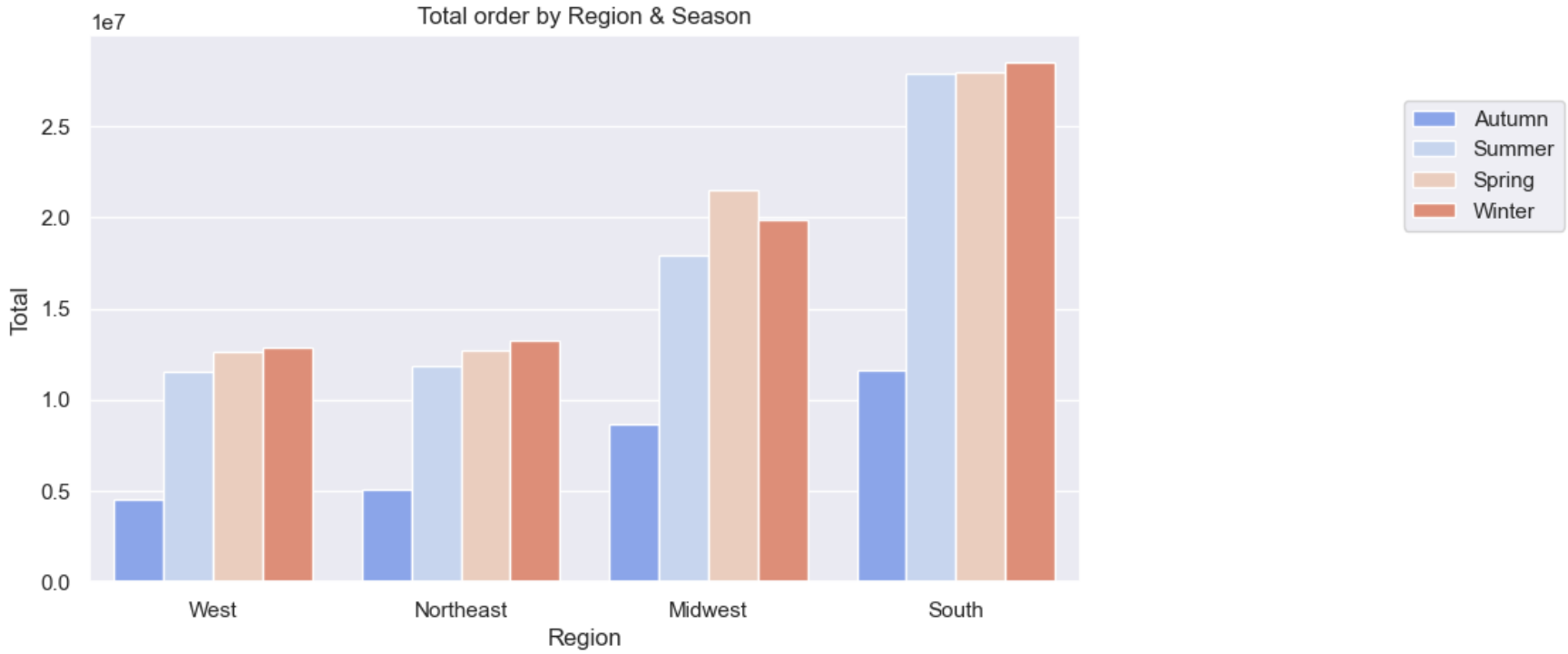
In [150]:
```
region23 = df.groupby(['Region', 'Season'])['Total'].sum().reset_index().sort_values(by='Total')
region23
```

Out[150]:

	Region	Season	Total
12	West	Autumn	4.531247e+06
4	Northeast	Autumn	5.089094e+06
0	Midwest	Autumn	8.692835e+06
14	West	Summer	1.150610e+07
8	South	Autumn	1.158153e+07
6	Northeast	Summer	1.182091e+07
13	West	Spring	1.264812e+07
5	Northeast	Spring	1.267823e+07
15	West	Winter	1.284360e+07
7	Northeast	Winter	1.322239e+07
2	Midwest	Summer	1.791172e+07
3	Midwest	Winter	1.987913e+07
1	Midwest	Spring	2.151355e+07
10	South	Summer	2.788650e+07
9	South	Spring	2.794032e+07
11	South	Winter	2.853956e+07

In [151]:

```
plt.figure(figsize=(9,5))
plt.title('Total order by Region & Season')
ax = sns.barplot(x='Region',y='Total', data=region23, hue='Season', palette='coolwarm', ci=None)
ax.legend(loc='upper right',bbox_to_anchor=(1.5, 0.9));
```



Insight: Autum months appears to b the lowest for all Total orders as everyother season seems to be equaly distributed

In [184]:
```
# FINALLY, lets Checks the Highest and most regular customer(Top_10) with how many times they made the orders
df[['UserName','Category', 'State', 'Region', 'County','CustomerSince','Cus_years_Loyalty']].\
value_counts().head(10).reset_index()
```

Out[184]:

	UserName	Category	State	Region	County	CustomerSince	Cus_years_Loyalty	0
0	jugonzalez	Health & Sports	IL	Midwest	DeKalb	2005-11-30	GenZ_Cus_22	2524
1	exbailes	Men's Fashion	PA	Northeast	Armstrong	2017-02-08	Newest_Cus_6	408
2	hmbeebe	Mobiles & Tablets	VT	Northeast	Rutland	2001-06-13	GenZ_Cus_22	397
3	lumelo	Men's Fashion	CO	West	Sedgwick	2015-05-17	Recent_Cus_12	303
4	exbailes	Mobiles & Tablets	PA	Northeast	Armstrong	2017-02-08	Newest_Cus_6	281
5	iobanner	Others	IA	Midwest	Page	2015-07-25	Recent_Cus_12	258
6	sahohn	Superstore	OH	Midwest	Seneca	2017-03-31	Newest_Cus_6	255
7	pnbraddy	Home & Living	TX	South	El Paso	2016-12-25	Newest_Cus_6	231
8	lumelo	Soghaat	CO	West	Sedgwick	2015-05-17	Recent_Cus_12	204
9	gebhatt	Mobiles & Tablets	IN	Midwest	Newton	2004-03-08	GenZ_Cus_22	198

In [178]:
```
# Further checking the Category of Product that had the maximum Total order
MaxTotal_order = df.loc[df['Total'] == df['Total'].max(), ['Total','Category','UserName',\
                                                           'State','Region','County', 'Cus_years_Loyalty']]
MaxTotal_order
```

Out[178]:

	Total	Category	UserName	State	Region	County	Cus_years_Loyalty
38265	101262.59	Beauty & Grooming	jmdavison	SD	Midwest	Lawrence	Newest_Cus_6

In [153]:
```
df[df['UserName'] == 'jugonzalez' ]
```

Out[153]:

QtyOrdered	Total	Year_1st_order	Year	Month	Month_name	Day_name	Quarter	Cus_years	Season	Cus_years_Loyalty	Region_State	I
2	13.3	2005	2022	3	March	Thursday	1	GenZ_Cus_22	Spring	GenZ_Cus_22	Midwest-IL	
2	13.3	2005	2022	3	March	Thursday	1	GenZ_Cus_22	Spring	GenZ_Cus_22	Midwest-IL	
2	13.3	2005	2022	3	March	Thursday	1	GenZ_Cus_22	Spring	GenZ_Cus_22	Midwest-IL	
2	16.0	2005	2022	3	March	Thursday	1	GenZ_Cus_22	Spring	GenZ_Cus_22	Midwest-IL	
2	26.6	2005	2022	3	March	Thursday	1	GenZ_Cus_22	Spring	GenZ_Cus_22	Midwest-IL	
...	...	...	...	...	...	...	...	...	...	...	...	
2	13.3	2005	2022	8	August	Saturday	3	GenZ_Cus_22	Summer	GenZ_Cus_22	Midwest-IL	
2	13.3	2005	2022	8	August	Saturday	3	GenZ_Cus_22	Summer	GenZ_Cus_22	Midwest-IL	
2	16.0	2005	2022	8	August	Saturday	3	GenZ_Cus_22	Summer	GenZ_Cus_22	Midwest-IL	
2	18.6	2005	2022	8	August	Saturday	3	GenZ_Cus_22	Summer	GenZ_Cus_22	Midwest-IL	
2	19.5	2005	2022	8	August	Saturday	3	GenZ_Cus_22	Summer	GenZ_Cus_22	Midwest-IL	

Final Insight and Recommendations

- The Sales are mostly in the Southern Region followed by Midwestern region, having the Northeast and West as the least areas for customer order
- The winter and the spring are the seasons where most orders do come

- There is no clear distinctive difference between customer Gender and they are almost equal with subtle differences both in count and Total orders
- Payment method mostly used are cod and easy pay, although most Total orders were returned from easypay, bankafalah and easypayvoucher
- On the category of items sold, the Mobile& Tablets, men's fashion and appliances are highest in numbers, however, on the total orders, entertainment and computing together with mobile and tables are mostly ordered.
- Advanced and Adult customer orders are declining, perhaps this should be a recommendation to stock more items used by aging people.

RECOMMENDATIONS

- Sterling E-commerce should consider collecting data on the cost of items to be able to determine the profit margins on each products
- Because of the reason above, we could not estimate if the items on sales are returning profit or loss
- We also recommend stocking old people items as we noticed a declining order for Advanced and Elderly Customers
- Perhaps also having a customer loyalty point system in their business as we noticed some usernames which are reoccurring in the dataset, but could not explore more on them (e.g. Jugonzalez ordered 2,524 times between 2021 and 2022
- Expanding to High-Performing Regions: The region with the largest orders come from the South, followed by the Midwest and the least come from the West. Marketing Campaigns
- Enhancing Customer Engagement: The highly loyal customers and the ones with a long-term relationship with the company could be offered personalized rewards, incentives, or exclusive benefit