In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df_customer = pd.read_csv(r'C:\Users\Käyttäjä\OneDrive - LUT University\Documents 1
```

In [3]:
```
df_transaction = pd.read_excel(r'C:\Users\Käyttäjä\OneDrive - LUT University\Docume
```

In [4]:
```
df_customer
```

Out[4]:

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	YOUNG SINGLES/COUPLES	Premium
1	1002	YOUNG SINGLES/COUPLES	Mainstream
2	1003	YOUNG FAMILIES	Budget
3	1004	OLDER SINGLES/COUPLES	Mainstream
4	1005	MIDAGE SINGLES/COUPLES	Mainstream
...	...	...	...
72632	2370651	MIDAGE SINGLES/COUPLES	Mainstream
72633	2370701	YOUNG FAMILIES	Mainstream
72634	2370751	YOUNG FAMILIES	Premium
72635	2370961	OLDER FAMILIES	Budget
72636	2373711	YOUNG SINGLES/COUPLES	Mainstream

72637 rows × 3 columns

In [5]:
```
df_customer.isnull().sum()
```

Out[5]:
```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64
```

In [6]:
```
df_customer['LIFESTAGE'].unique()
```

Out[6]:
```
array(['YOUNG SINGLES/COUPLES', 'YOUNG FAMILIES', 'OLDER SINGLES/COUPLES',
       'MIDAGE SINGLES/COUPLES', 'NEW FAMILIES', 'OLDER FAMILIES',
       'RETIREES'], dtype=object)
```

In [7]:
```
df_customer['PREMIUM_CUSTOMER'].unique()
```

Out[7]:
```
array(['Premium', 'Mainstream', 'Budget'], dtype=object)
```

In [8]:
```
df_customer.duplicated().any()
```

Out[8]: False

In [9]:
```
df_transaction
```

Out[9]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD
0	43390	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	43599	1	1307	348	66	CCs Nacho Cheese 175g	
2	43605	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
3	43329	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	
4	43330	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	
...	...	...	...	...	...	...	
264831	43533	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	
264832	43325	272	272358	270154	74	Tostitos Splash Of Lime 175g	
264833	43410	272	272379	270187	51	Doritos Mexicana 170g	
264834	43461	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	
264835	43365	272	272380	270189	74	Tostitos Splash Of Lime 175g	

264836 rows × 8 columns

In [10]:
```
df_transaction.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 8 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   DATE            264836 non-null  int64
 1   STORE_NBR       264836 non-null  int64
 2   LYLTY_CARD_NBR  264836 non-null  int64
 3   TXN_ID          264836 non-null  int64
 4   PROD_NBR        264836 non-null  int64
 5   PROD_NAME       264836 non-null  object
 6   PROD_QTY        264836 non-null  int64
 7   TOT_SALES       264836 non-null  float64
dtypes: float64(1), int64(6), object(1)
memory usage: 16.2+ MB
```

In [15]:
```
df_transaction.duplicated().any()
```

Out[15]: False

In [13]:
```
df_transaction[df_transaction.duplicated()]
```

Out[13]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_C
124845	2018-10-01	107	107024	108462	45	Smiths Thinly Cut Roast Chicken 175g	

In [14]:
```
df_transaction = df_transaction.drop_duplicates()
```

In [11]:
```
 # change type of date to date using  pd.to_datetime() function '
#('1899-12-30' base date) and the unit as 'D' (days)
df_transaction['DATE'] = pd.to_datetime( df_transaction['DATE'], origin='1899-12-30
```

In [16]:
```
sorted(df_transaction['PROD_NAME'].unique())
```

Out[16]:
```
['Burger Rings 220g',
 'CCs Nacho Cheese    175g',
 'CCs Original 175g',
 'CCs Tasty Cheese    175g',
 'Cheetos Chs & Bacon Balls 190g',
 'Cheetos Puffs 165g',
 'Cheezels Cheese 330g',
 'Cheezels Cheese Box 125g',
 'Cobs Popd Sea Salt  Chips 110g',
 'Cobs Popd Sour Crm  &Chives Chips 110g',
 'Cobs Popd Swt/Chlli &Sr/Cream Chips 110g',
 'Dorito Corn Chp     Supreme 380g',
 'Doritos Cheese      Supreme 330g',
 'Doritos Corn Chip Mexican Jalapeno 150g',
 'Doritos Corn Chip Southern Chicken 150g',
 'Doritos Corn Chips  Cheese Supreme 170g',
 'Doritos Corn Chips  Nacho Cheese 170g',
 'Doritos Corn Chips  Original 170g',
 'Doritos Mexicana    170g',
 'Doritos Salsa       Medium 300g',
 'Doritos Salsa Mild  300g',
 'French Fries Potato Chips 175g',
 'Grain Waves         Sweet Chilli 210g',
 'Grain Waves Sour    Cream&Chives 210G',
 'GrnWves Plus Btroot & Chilli Jam 180g',
 'Infuzions BBQ Rib   Prawn Crackers 110g',
 'Infuzions Mango     Chutny Papadums 70g',
 'Infuzions SourCream&Herbs Veg Strws 110g',
 'Infuzions Thai SweetChili PotatoMix 110g',
 'Infzns Crn Crnchers Tangy Gcamole 110g',
 'Kettle 135g Swt Pot Sea Salt',
 'Kettle Chilli 175g',
 'Kettle Honey Soy    Chicken 175g',
 'Kettle Mozzarella   Basil & Pesto 175g',
 'Kettle Original 175g',
 'Kettle Sea Salt     And Vinegar 175g',
 'Kettle Sensations   BBQ&Maple 150g',
 'Kettle Sensations   Camembert & Fig 150g',
 'Kettle Sensations   Siracha Lime 150g',
 'Kettle Sweet Chilli And Sour Cream 175g',
 'Kettle Tortilla ChpsBtroot&Ricotta 150g',
 'Kettle Tortilla ChpsFeta&Garlic 150g',
 'Kettle Tortilla ChpsHny&Jlpno Chili 150g',
 'NCC Sour Cream &    Garden Chives 175g',
 'Natural Chip        Compny SeaSalt175g',
 'Natural Chip Co     Tmato Hrb&Spce 175g',
 'Natural ChipCo      Hony Soy Chckn175g',
 'Natural ChipCo Sea  Salt & Vinegr 175g',
 'Old El Paso Salsa   Dip Chnky Tom Ht300g',
 'Old El Paso Salsa   Dip Tomato Med 300g',
 'Old El Paso Salsa   Dip Tomato Mild 300g',
 'Pringles Barbeque   134g',
 'Pringles Chicken    Salt Crips 134g',
 'Pringles Mystery    Flavour 134g',
 'Pringles Original   Crisps 134g',
 'Pringles Slt Vingar 134g',
```

```
'Pringles SourCream  Onion 134g',
'Pringles Sthrn FriedChicken 134g',
'Pringles Sweet&Spcy BBQ 134g',
'RRD Chilli&         Coconut 150g',
'RRD Honey Soy       Chicken 165g',
'RRD Lime & Pepper   165g',
'RRD Pc Sea Salt     165g',
'RRD SR Slow Rst     Pork Belly 150g',
'RRD Salt & Vinegar  165g',
'RRD Steak &         Chimuchurri 150g',
'RRD Sweet Chilli &  Sour Cream 165g',
'Red Rock Deli Chikn&Garlic Aioli 150g',
'Red Rock Deli SR    Salsa & Mzzrlla 150g',
'Red Rock Deli Sp    Salt & Truffle 150G',
'Red Rock Deli Thai  Chilli&Lime 150g',
'Smith Crinkle Cut   Bolognese 150g',
'Smith Crinkle Cut   Mac N Cheese 150g',
'Smiths Chip Thinly  Cut Original 175g',
'Smiths Chip Thinly  CutSalt/Vinegr175g',
'Smiths Chip Thinly  S/Cream&Onion 175g',
'Smiths Crinkle      Original 330g',
'Smiths Crinkle Chips Salt & Vinegar 330g',
'Smiths Crinkle Cut  Chips Barbecue 170g',
'Smiths Crinkle Cut  Chips Chicken 170g',
'Smiths Crinkle Cut  Chips Chs&Onion170g',
'Smiths Crinkle Cut  Chips Original 170g',
'Smiths Crinkle Cut  French OnionDip 150g',
'Smiths Crinkle Cut  Salt & Vinegar 170g',
'Smiths Crinkle Cut  Snag&Sauce 150g',
'Smiths Crinkle Cut  Tomato Salsa 150g',
'Smiths Crnkle Chip  Orgnl Big Bag 380g',
'Smiths Thinly       Swt Chli&S/Cream175G',
'Smiths Thinly Cut   Roast Chicken 175g',
'Snbts Whlgrn Crisps Cheddr&Mstrd 90g',
'Sunbites Whlegrn    Crisps Frch/Onin 90g',
'Thins Chips         Originl saltd 175g',
'Thins Chips Light&  Tangy 175g',
'Thins Chips Salt &  Vinegar 175g',
'Thins Chips Seasonedchicken 175g',
'Thins Potato Chips  Hot & Spicy 175g',
'Tostitos Lightly    Salted 175g',
'Tostitos Smoked     Chipotle 175g',
'Tostitos Splash Of  Lime 175g',
'Twisties Cheese     270g',
'Twisties Cheese     Burger 250g',
'Twisties Chicken270g',
'Tyrrells Crisps     Ched & Chives 165g',
'Tyrrells Crisps     Lightly Salted 165g',
'WW Crinkle Cut      Chicken 175g',
'WW Crinkle Cut      Original 175g',
'WW D/Style Chip     Sea Salt 200g',
'WW Original Corn    Chips 200g',
'WW Original Stacked Chips 160g',
'WW Sour Cream &OnionStacked Chips 160g',
'WW Supreme Cheese   Corn Chips 200g',
'Woolworths Cheese   Rings 190g',
```

```
 'Woolworths Medium   Salsa 300g',
 'Woolworths Mild     Salsa 300g']
```

In [22]:
```
# List of product names to remove which is not chips product
product_names_to_remove = [
    'Woolworths Medium   Salsa 300g',
    'Woolworths Mild     Salsa 300g',
    'Old El Paso Salsa   Dip Chnky Tom Ht300g',
    'Old El Paso Salsa   Dip Tomato Med 300g',
    'Old El Paso Salsa   Dip Tomato Mild 300g',
    'Doritos Salsa       Medium 300g',
    'Doritos Salsa Mild  300g'
]

# Filter out rows where PROD_NAME is in the list
df_transaction = df_transaction[~df_transaction['PROD_NAME'].isin(product_names_to_
```

In [23]:
```
import re

def extract_brand_and_size(prod_name):
    """
    Extracts the brand (first word by default) and size (e.g., '175g') from the pro
    """
    # Extract brand as the first word
    brand = prod_name.split()[0]

    # Look for size pattern (e.g., '175g', '200G', etc.)
    size_match = re.search(r'\d+\s*[gG]', prod_name)
    size = size_match.group() if size_match else None  # If no match, return None

    return brand, size

# Apply the function to the 'PROD_NAME' column and create new columns
df_transaction[['Brand', 'Size']] = df_transaction['PROD_NAME'].apply(
    lambda x: pd.Series(extract_brand_and_size(x))
)

# Display the updated DataFrame
print(df_transaction[['PROD_NAME', 'Brand', 'Size']])
```

```
                                     PROD_NAME     Brand  Size
0         Natural Chip        Compny SeaSalt175g   Natural  175g
1                       CCs Nacho Cheese    175g       CCs  175g
2         Smiths Crinkle Cut  Chips Chicken 170g    Smiths  170g
3         Smiths Chip Thinly  S/Cream&Onion 175g    Smiths  175g
4       Kettle Tortilla ChpsHny&Jlpno Chili 150g    Kettle  150g
...                                          ...       ...   ...
264831   Kettle Sweet Chilli And Sour Cream 175g    Kettle  175g
264832             Tostitos Splash Of  Lime 175g  Tostitos  175g
264833                  Doritos Mexicana    170g   Doritos  170g
264834   Doritos Corn Chip Mexican Jalapeno 150g   Doritos  150g
264835             Tostitos Splash Of  Lime 175g  Tostitos  175g

[249669 rows x 3 columns]
```

In [24]:
```
sorted(df_transaction['Brand'].unique())
```

Out[24]:
```
['Burger',
 'CCs',
 'Cheetos',
 'Cheezels',
 'Cobs',
 'Dorito',
 'Doritos',
 'French',
 'Grain',
 'GrnWves',
 'Infuzions',
 'Infzns',
 'Kettle',
 'NCC',
 'Natural',
 'Pringles',
 'RRD',
 'Red',
 'Smith',
 'Smiths',
 'Snbts',
 'Sunbites',
 'Thins',
 'Tostitos',
 'Twisties',
 'Tyrrells',
 'WW',
 'Woolworths']
```

In [26]:
```
# Define a dictionary with old values as keys and new values as values
replace_dict = {
    'Burger': 'Burger Ring',
    'Dorito': 'Doritos',
    'French': 'French Fries',
    'Grain':'Grain Waves',
    'GrnWves': 'Grain Waves',
    'Infzns': 'Infuzions',
    'Red': 'Red Rock Deli',
    'RRD': 'Red Rock Deli',
    'Smith': 'Smiths',
    'Snbts': 'Sunbites Whlegrn',
    'NCC': 'Natural Chip Co',
    'Natural': 'Natural Chip Co',
    'Sunbites':'Sunbites Whlegrn',
    'WW': 'Woolworths'

}

# Replace values in the 'PROD_NAME' column based on the dictionary
df_transaction['Brand'] = df_transaction['Brand'].replace(replace_dict)

# Check the modified dataframe
print(sorted(df_transaction['Brand'].unique()))
```

```
['Burger Ring', 'CCs', 'Cheetos', 'Cheezels', 'Cobs', 'Doritos', 'French Fries', 'Gr
ain Waves', 'Infuzions', 'Kettle', 'Natural Chip Co', 'Pringles', 'Red Rock Deli',
'Smiths', 'Sunbites Whlegrn', 'Thins', 'Tostitos', 'Twisties', 'Tyrrells', 'Woolwort
hs']
```

In [86]:
```
sorted(df_transaction['Size'].unique())
```

Out[86]:
```
[70,
 90,
 110,
 125,
 134,
 135,
 150,
 160,
 165,
 170,
 175,
 180,
 190,
 200,
 210,
 220,
 250,
 270,
 330,
 380]
```

In [76]:
```
df_transaction['Size'] = df_transaction['Size'].str.replace(r'[gG]', '', regex=True
```

In [85]:
```
df_transaction['Size'] = df_transaction['Size'].astype(int)
```

In [80]:
```
df_transaction.describe()
```

Out[80]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR
count	249667	249667.000000	2.496670e+05	2.496670e+05	249667.000000
mean	2018-12-30 02:26:13.683346176	135.043662	1.355197e+05	1.351228e+05	56.294753
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.000000
25%	2018-09-30 00:00:00	70.000000	7.001600e+04	6.757350e+04	27.000000
50%	2018-12-30 00:00:00	130.000000	1.303600e+05	1.351470e+05	53.000000
75%	2019-03-31 00:00:00	203.000000	2.030790e+05	2.026325e+05	86.000000
max	2019-06-30 00:00:00	272.000000	2.373711e+06	2.415841e+06	114.000000
std	NaN	76.773600	8.065751e+04	7.813158e+04	33.528625

In [81]:
```
# we notice there outlier in  product quantity, and TOT_SALES  which we should inve
df_transaction[df_transaction['PROD_QTY'] == 200.000000]
```

Out[81]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TO

In [37]:
```
#there  two transactions where PROD_QTY = 200 and TOT_SALES = 650 with samw custome
#let's see if totalsales outliers with same transcation
[dfdf_transaction_transaction['TOT_SALES'] == 650.0]
```

Out[37]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QT
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	2(
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	2(

In [39]:
```
#same case where customer 226000 bought same quantity. let see if he other purchase
df_transaction[df_transaction['LYLTY_CARD_NBR'] == 226000]
```

Out[39]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QT
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	2(
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	2(

In [41]:
```
#He has only two previous transactions, and his purchases are different from others
#We could say it is better to remove the outliers and remove rows related to custom
df_transaction = df_transaction[df_transaction['LYLTY_CARD_NBR'] != 226000]
df_transaction.describe()
```

Out[41]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR
count	249667	249667.000000	2.496670e+05	2.496670e+05	249667.000000
mean	2018-12-30 02:26:13.683346176	135.043662	1.355197e+05	1.351228e+05	56.294753
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.000000
25%	2018-09-30 00:00:00	70.000000	7.001600e+04	6.757350e+04	27.000000
50%	2018-12-30 00:00:00	130.000000	1.303600e+05	1.351470e+05	53.000000
75%	2019-03-31 00:00:00	203.000000	2.030790e+05	2.026325e+05	86.000000
max	2019-06-30 00:00:00	272.000000	2.373711e+06	2.415841e+06	114.000000
std	NaN	76.773600	8.065751e+04	7.813158e+04	33.528625

In [87]:
```
# Merge using a left join
merged_df = pd.merge(df_transaction, df_customer, on='LYLTY_CARD_NBR', how='left')
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 249667 entries, 0 to 249666
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype         
---  ------            --------------   -----         
 0   DATE              249667 non-null  datetime64[ns]
 1   STORE_NBR         249667 non-null  int64         
 2   LYLTY_CARD_NBR    249667 non-null  int64         
 3   TXN_ID            249667 non-null  int64         
 4   PROD_NBR          249667 non-null  int64         
 5   PROD_NAME         249667 non-null  object        
 6   PROD_QTY          249667 non-null  int64         
 7   TOT_SALES         249667 non-null  float64       
 8   Brand             249667 non-null  object        
 9   Size              249667 non-null  int32         
 10  LIFESTAGE         249667 non-null  object        
 11  PREMIUM_CUSTOMER  249667 non-null  object        
dtypes: datetime64[ns](1), float64(1), int32(1), int64(5), object(4)
memory usage: 21.9+ MB
```

In [43]:
```
# List of related columns
columns_to_select = ['DATE', 'STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR', '
                     ,'Brand', 'Size','TOT_SALES']

# Selecting only the desired columns from df_transaction
merged_df = merged_df[columns_to_select]
```

In [83]:
```
merged_df
```

Out[83]:

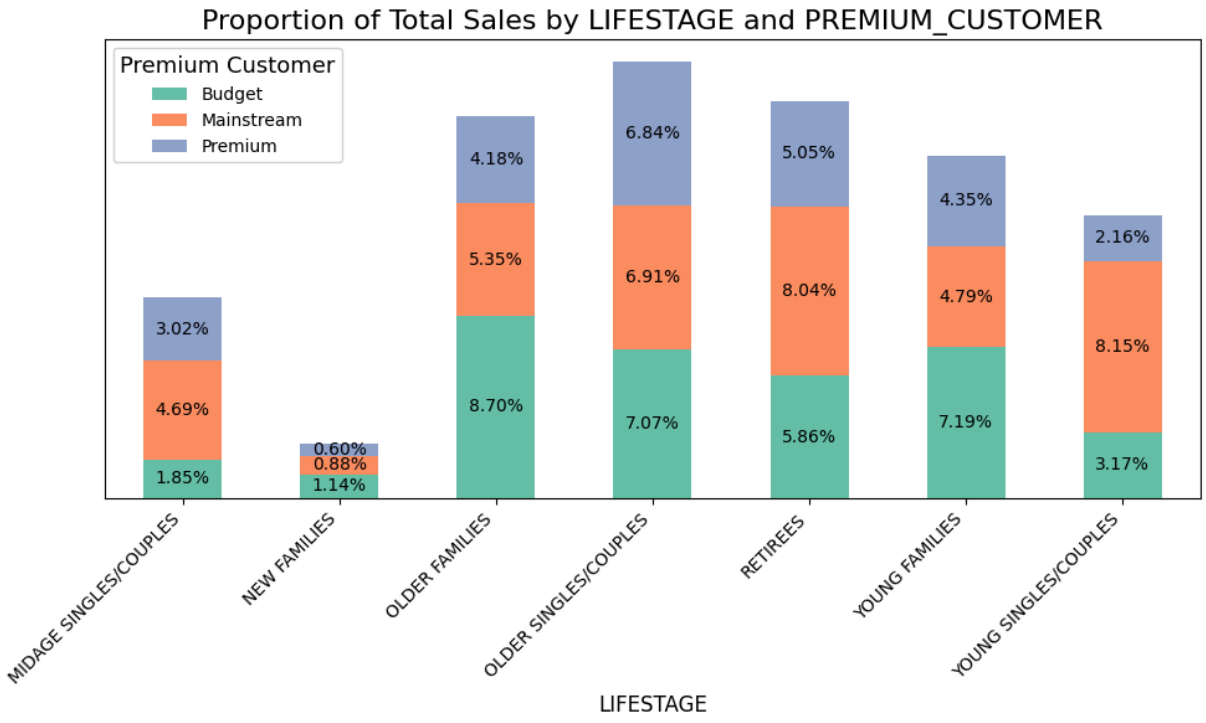
	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD
0	2018-10-17	1	1000	1	5	Natural Chip Compny SeaSalt175g	
1	2019-05-14	1	1307	348	66	CCs Nacho Cheese 175g	
2	2019-05-20	1	1343	383	61	Smiths Crinkle Cut Chips Chicken 170g	
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	
4	2018-08-18	2	2426	1038	108	Kettle Tortilla ChpsHny&Jlpno Chili 150g	
...	...	...	...	...	...	...	
249662	2019-03-09	272	272319	270088	89	Kettle Sweet Chilli And Sour Cream 175g	
249663	2018-08-13	272	272358	270154	74	Tostitos Splash Of Lime 175g	
249664	2018-11-06	272	272379	270187	51	Doritos Mexicana 170g	
249665	2018-12-27	272	272379	270188	42	Doritos Corn Chip Mexican Jalapeno 150g	
249666	2018-09-22	272	272380	270189	74	Tostitos Splash Of Lime 175g	

249667 rows × 12 columns

In [47]:
```
#Data analysis on customer segments
#frame question :
#customer segments
#What are the total sales for each customer segment?
# how many customers are in each segment?
# how many purchase are in each segment?
```

In [59]:
```
#What are the total sales for each customer segment?
# Step 1: Calculate total sales by LIFESTAGE and PREMIUM_CUSTOMER
sales_by_segment = merged_df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['TOT_SALES'
```

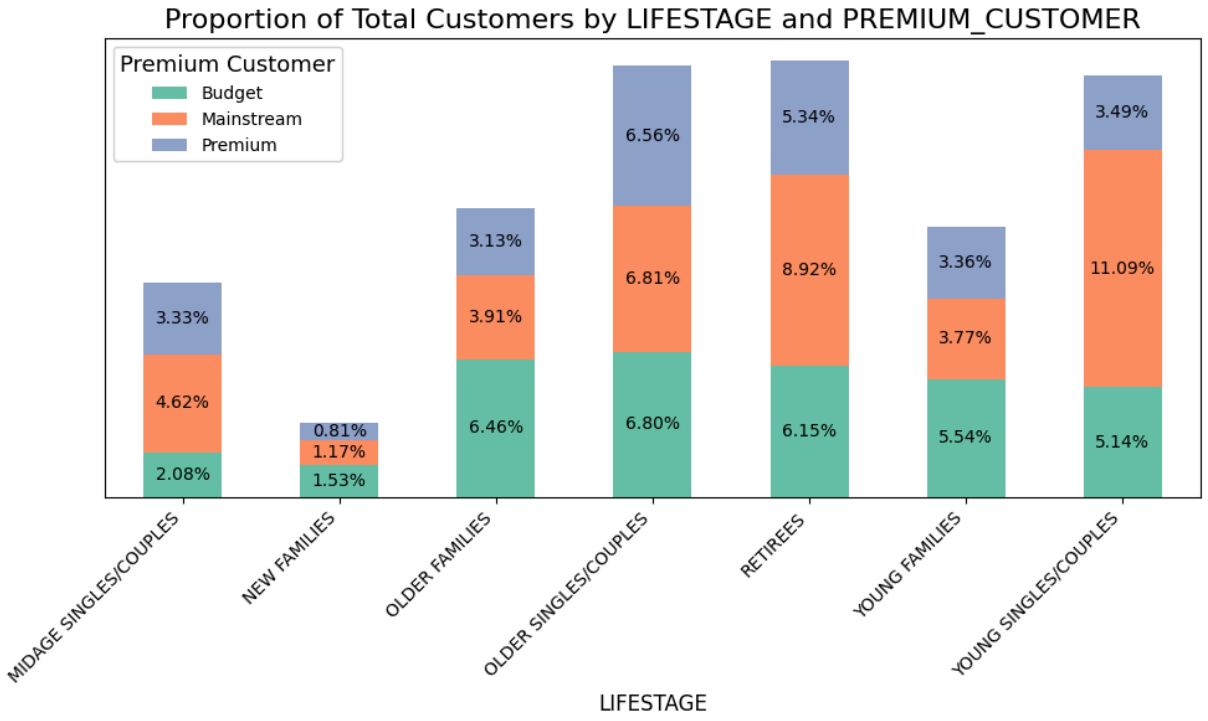
```
# Step 2: Calculate total sales across all segments (this is the total for the enti
total_sales = sales_by_segment['TOT_SALES'].sum()

# Step 3: Calculate the proportion of total sales for each LIFESTAGE and PREMIUM_CU
sales_by_segment['Proportion of Total Sales'] = sales_by_segment['TOT_SALES'] / tot

# Step 4: Create a pivot table to reshape the data (LIFESTAGE vs PREMIUM_CUSTOMER)
sales_pivot = sales_by_segment.pivot_table(index='LIFESTAGE', columns='PREMIUM_CUST
                                           values='Proportion of Total Sales', aggf

# Step 5: Plotting the stacked column chart with custom colors and adding labels
fig, ax = plt.subplots(figsize=(10, 6))

# Custom color palette for PREMIUM_CUSTOMER (you can modify the colors here)
colors = sns.color_palette("Set2", n_colors=sales_pivot.shape[1])

# Plot the stacked bar chart
sales_pivot.plot(kind='bar', stacked=True, ax=ax, color=colors)

# Add labels on top of each stack segment
for p in ax.patches:
    height = p.get_height()
    width = p.get_width()
    x = p.get_x() + width / 2  # Position the label in the center of the bar
    y = p.get_y() + height / 2  # Position the label in the center of the stack

    # Display the label (percentage of total sales)
    ax.text(x, y, f'{height:.2%}', ha='center', va='center', fontsize=10, color='bl

# Add labels and title
plt.title('Proportion of Total Sales by LIFESTAGE and PREMIUM_CUSTOMER', fontsize=1
plt.xlabel('LIFESTAGE', fontsize=12)

# Hide y-axis
ax.yaxis.set_visible(False)

# Rotate x-axis labels for better visibility
plt.xticks(rotation=45, ha='right')

# Add legend title
plt.legend(title='Premium Customer', title_fontsize='13')

# Show the plot
plt.tight_layout()
plt.show()
```



In [60]:
```
# the top 3 segments  contributing the most to total sales are Budget - older famil
# Mainstream - young singles/couples, and Mainstream- retirees
```

In [61]:
```
## how many customers are in each segment?


# Step 1: Calculate the number of customers by LIFESTAGE and PREMIUM_CUSTOMER
cus_by_segment = merged_df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER'])['LYLTY_CARD_N

# Step 2: Calculate the total number of customers across all segments
total_cus = cus_by_segment['LYLTY_CARD_NBR'].sum()

# Step 3: Calculate the proportion of customers in each segment
cus_by_segment['Proportion of Total cus'] = cus_by_segment['LYLTY_CARD_NBR'] / tota

# Step 4: Create a pivot table to reshape the data (LIFESTAGE vs PREMIUM_CUSTOMER)
cus_pivot = cus_by_segment.pivot_table(index='LIFESTAGE', columns='PREMIUM_CUSTOMER
                                       values='Proportion of Total cus', aggfun

# Step 5: Plotting the stacked column chart with custom colors and adding labels
fig, ax = plt.subplots(figsize=(10, 6))

# Custom color palette for PREMIUM_CUSTOMER (you can modify the colors here)
colors = sns.color_palette("Set2", n_colors=cus_pivot.shape[1])

# Plot the stacked bar chart
cus_pivot.plot(kind='bar', stacked=True, ax=ax, color=colors)

# Add labels on top of each stack segment
for p in ax.patches:
    height = p.get_height()
    width = p.get_width()
    x = p.get_x() + width / 2  # Position the label in the center of the bar
```

```
    y = p.get_y() + height / 2  # Position the label in the center of the stack

    # Display the label (percentage of total customers)
    ax.text(x, y, f'{height:.2%}', ha='center', va='center', fontsize=10, color='bl

# Add labels and title
plt.title('Proportion of Total Customers by LIFESTAGE and PREMIUM_CUSTOMER', fontsi
plt.xlabel('LIFESTAGE', fontsize=12)

# Hide y-axis
ax.yaxis.set_visible(False)

# Rotate x-axis labels for better visibility
plt.xticks(rotation=45, ha='right')

# Add legend title
plt.legend(title='Premium Customer', title_fontsize='13')

# Show the plot
plt.tight_layout()
plt.show()
```



In [63]:
```
#Mainstream - Young Singles/Couples  have the largest customer numbers, which means
#Budget - Older Families has high total sales, but the number of customers in this
#is relatively smaller (6.46% of the total customer base).
#This insight suggests that:
#Budget - Older Families may have a higher average spend per customer, making this
#in terms of sales despite having fewer customers.
#To better understand the relationship between total sales and number of customers,
#we can calculate the average sales per customer for each segment.

#How to Calculate Average Sales per Customer:
#  we have the total sales and number of unique customers by LIFESTAGE and PREMIUM_
#sales_by_segment, cus_by_segment
```

```
# Step 1: Merge the two dataframes to get total sales and number of customers in on
merged_segment = pd.merge(sales_by_segment, cus_by_segment, on=['LIFESTAGE', 'PREMI

# Step 2: Calculate average sales per customer
merged_segment['Avg_Sales_Per_Customer'] = merged_segment['TOT_SALES'] / merged_seg

# Step 3: View the segments with the highest average sales per customer
merged_segment = merged_segment.sort_values(by='Avg_Sales_Per_Customer', ascending=

# Display the top 10 segments with the highest average sales per customer
print(merged_segment[['LIFESTAGE', 'PREMIUM_CUSTOMER', 'Avg_Sales_Per_Customer']])
```

	LIFESTAGE	PREMIUM_CUSTOMER	Avg_Sales_Per_Customer
7	OLDER FAMILIES	Mainstream	34.792865
6	OLDER FAMILIES	Budget	34.266540
8	OLDER FAMILIES	Premium	33.920982
15	YOUNG FAMILIES	Budget	33.052020
17	YOUNG FAMILIES	Premium	32.965516
16	YOUNG FAMILIES	Mainstream	32.390587
11	OLDER SINGLES/COUPLES	Premium	26.514071
9	OLDER SINGLES/COUPLES	Budget	26.456373
1	MIDAGE SINGLES/COUPLES	Mainstream	25.821548
10	OLDER SINGLES/COUPLES	Mainstream	25.818706
12	RETIREES	Budget	24.250728
14	RETIREES	Premium	24.064891
2	MIDAGE SINGLES/COUPLES	Premium	23.117325
13	RETIREES	Mainstream	22.928353
0	MIDAGE SINGLES/COUPLES	Budget	22.712534
4	NEW FAMILIES	Mainstream	19.278177
3	NEW FAMILIES	Budget	18.970742
5	NEW FAMILIES	Premium	18.824437
19	YOUNG SINGLES/COUPLES	Mainstream	18.705826
20	YOUNG SINGLES/COUPLES	Premium	15.745458
18	YOUNG SINGLES/COUPLES	Budget	15.679565

In [68]:
```
#Older Families tend to spend significantly more on average compared to other custo
#multi-pack buying behavior
# let's investigate if customers are purchasing more than one unit of chips in a si
# Group by LIFESTAGE and PREMIUM_CUSTOMER to analyze average quantity
avg_qty_segment_analysis = merged_df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).agg
    avg_qty=('PROD_QTY', 'mean')
).reset_index()

# Sort by average quantity in descending order for better readability
avg_qty_segment_analysis = avg_qty_segment_analysis.sort_values(by='avg_qty', ascen

# Display the result
avg_qty_segment_analysis
```

Out[68]:

	LIFESTAGE	PREMIUM_CUSTOMER	avg_qty
7	OLDER FAMILIES	Mainstream	1.948550
8	OLDER FAMILIES	Premium	1.945572
6	OLDER FAMILIES	Budget	1.945387
15	YOUNG FAMILIES	Budget	1.941428
16	YOUNG FAMILIES	Mainstream	1.941259
17	YOUNG FAMILIES	Premium	1.937540
9	OLDER SINGLES/COUPLES	Budget	1.914327
11	OLDER SINGLES/COUPLES	Premium	1.913984
1	MIDAGE SINGLES/COUPLES	Mainstream	1.911875
10	OLDER SINGLES/COUPLES	Mainstream	1.911206
14	RETIREES	Premium	1.901116
12	RETIREES	Budget	1.893066
0	MIDAGE SINGLES/COUPLES	Budget	1.892363
2	MIDAGE SINGLES/COUPLES	Premium	1.891130
13	RETIREES	Mainstream	1.886645
5	NEW FAMILIES	Premium	1.861406
4	NEW FAMILIES	Mainstream	1.856366
3	NEW FAMILIES	Budget	1.855286
19	YOUNG SINGLES/COUPLES	Mainstream	1.852778
20	YOUNG SINGLES/COUPLES	Premium	1.806152
18	YOUNG SINGLES/COUPLES	Budget	1.806032

In [102…

```
#This analysis highlights that Older Families across all premium customer categorie
#(Budget, Mainstream, Premium) have the highest average quantity per transaction, w
#This strongly suggests that Older Families are likely engaging in multi-pack buyin
#now let's focus on deep diving into specific customer segments to gather more insi
#particularly on the Mainstream - Young Singles/Couples and Older Families (Budget)
#1. Mainstream - Young Singles/Couples
M_young_Singles_Couples = merged_df[(merged_df['LIFESTAGE']=='YOUNG SINGLES/COUPLES
M_young_Singles_Couples.groupby('Brand').agg( tot = ('TOT_SALES', 'sum' ),  qun= ('
```

Out[102]:

Brand	tot	qun
Kettle	35423.6	7172
Doritos	20925.9	4447
Pringles	16006.2	4326
Smiths	15265.7	3609
Infuzions	8749.4	2343
Twisties	7539.8	1673
Tostitos	7238.0	1645
Thins	7217.1	2187
Cobs	6144.6	1617
Red Rock Deli	4958.1	1753
Tyrrells	4800.6	1143
Grain Waves	4201.0	1185
Cheezels	3318.3	651
Natural Chip Co	2130.0	710
Woolworths	1605.8	873
Cheetos	898.8	291
CCs	850.5	405
French Fries	429.0	143
Sunbites Whlegrn	391.0	230
Burger Ring	243.8	106

In [115…

```
#Kettle chips dominate both in sales and quantity purchased, making it the top-perf
#It could indicate that Mainstream - Young Singles/Couples prefer this brand signif
#let figureout that with Analysis of Affinity to Brand:
#let understand how strongly Mainstream - Young Singles/Couples prefer a brand rela
 # Group by Brand to get total product quantity for the Mainstream - Young Singles/
brand_qty_segment = M_young_Singles_Couples.groupby('Brand').agg(
    segment_qty=('PROD_QTY', 'sum')
).reset_index()

# Group by Brand to get total product quantity for all customers
total_qty_all = merged_df.groupby('Brand').agg(
    total_qty_all=('PROD_QTY', 'sum')
).reset_index()

# Merge the two dataframes on 'Brand'
```

```
brand_qty_affinity = pd.merge(brand_qty_segment, total_qty_all, on='Brand', how='le

# Calculate affinity as the ratio of segment's quantity to total quantity
brand_qty_affinity['affinity'] = brand_qty_affinity['segment_qty'] / brand_qty_affi
brand_qty_affinity_sorted = brand_qty_affinity.sort_values(by='affinity', ascending
# Display the result
print(brand_qty_affinity_sorted)
```

	Brand	segment_qty	total_qty_all	affinity
18	Tyrrells	1143	12298	0.092942
17	Twisties	1673	18118	0.092339
5	Doritos	4447	48331	0.092011
9	Kettle	7172	79051	0.090726
16	Tostitos	1645	18134	0.090714
11	Pringles	4326	48019	0.090089
4	Cobs	1617	18571	0.087071
8	Infuzions	2343	27119	0.086397
15	Thins	2187	26929	0.081214
7	Grain Waves	1185	14726	0.080470
3	Cheezels	651	8747	0.074426
13	Smiths	3609	60337	0.059814
6	French Fries	143	2643	0.054105
2	Cheetos	291	5530	0.052622
12	Red Rock Deli	1753	33646	0.052101
10	Natural Chip Co	710	14106	0.050333
1	CCs	405	8609	0.047044
14	Sunbites Whlegrn	230	5692	0.040408
19	Woolworths	873	22333	0.039090
0	Burger Ring	106	2970	0.035690

In [121…

```
#Tyrrells, Twisties,Doritos, and  Kettle are the top 4 brands with the highest affi
#This means the Mainstream - Young Singles/Couples segment buys these brands more f
#than the overall customer base.
#The affinity scores for these brands range from 0.0929 to 0.092.
#Lower Affinity Brands:

#Burger Ring, CCs, and Woolworths have the lowest affinity scores,
#suggesting that these brands are less popular within the Mainstream - Young Single
#the total customer base.
#Now we want to looking for prefered size to Mainstream - Young Singles/Couples  re
 # Group by size to get total product quantity for the Mainstream - Young Singles/C
Size_qty_segment = M_young_Singles_Couples.groupby('Size').agg(
    segment_qty=('PROD_QTY', 'sum')
).reset_index()

# Group by Brand to get total product quantity for all customers
total_qty_alls = merged_df.groupby('Size').agg(
    total_qty_all=('PROD_QTY', 'sum')
).reset_index()

# Merge the two dataframes on 'Brand'
Size_qty_affinity = pd.merge(Size_qty_segment, total_qty_alls, on='Size', how='left

# Calculate affinity as the ratio of segment's quantity to total quantity
Size_qty_affinity['affinity'] = Size_qty_affinity['segment_qty'] / Size_qty_affinit
Size_qty_affinity_sorted = Size_qty_affinity.sort_values(by='affinity', ascending=F
```

```
# Display the result
print(Size_qty_affinity_sorted)
```

```
    Size  segment_qty  total_qty_all  affinity
17   270         1153          12049  0.095693
19   380         1165          12273  0.094924
18   330         2220          23999  0.092504
4    134         4326          48019  0.090089
2    110         3850          42835  0.089880
14   210         1055          11962  0.088196
5    135          535           6212  0.086124
16   250          520           6069  0.085681
9    170         2926          38088  0.076822
10   175         9237         126465  0.073040
6    150         5993          82174  0.072931
8    165         2016          29051  0.069395
12   190          271           5673  0.047770
11   180          130           2764  0.047033
7    160          232           5604  0.041399
1     90          230           5692  0.040408
3    125          109           2730  0.039927
13   200          325           8425  0.038576
0     70          110           2855  0.038529
15   220          106           2970  0.035690
```

In [125…

```
#The most popular pack size for Tyrrells in this segment is 270g with a quantity of
# let's find out which brands offer the most preferred sizes
M_young_Singles_Couples[M_young_Singles_Couples['Size'] == 270]['Brand'].unique()
```

Out[125]:

```
array(['Twisties'], dtype=object)
```

In [126…

```
#Twisties is the only brand has the size 270.
#and also it is the second preferd brand at young_Singles_Couples
# let drive to totalsale and quanity of Twisties
# Filter the data for Twisties in the Mainstream - Young Singles/Couples segment
twisties_data = M_young_Singles_Couples[M_young_Singles_Couples['Brand'] == 'Twisti

# Calculate the total sales and quantity for Twisties in the segment
twisties_sales = twisties_data.groupby('Size').agg(
    total_sales=('TOT_SALES', 'sum'),
    total_qty=('PROD_QTY', 'sum')
).reset_index()

# Display the result for Twisties sales by size
print(twisties_sales)
```

```
  Size  total_sales  total_qty
0  250       2236.0        520
1  270       5303.8       1153
```

Key Insights: The Mainstream - Young Singles/Couples segment has distinct preferences for certain brands and sizes, particularly Tyrrells 165g and Twisties 270g.

This segment has significant sales, making it a crucial target for future marketing, product placements, and promotions.

Recommendations for Management: Increase Visibility and Promotions for Tyrrells 165g: Focus on the Tyrrells 165g size as it is the top-performing pack in this segment. Capitalize on Twisties 270g: Use the popularity of Twisties 270g in marketing campaigns and stock it accordingly. Customer Engagement: Leverage targeted campaigns to maintain loyalty and encourage repeat purchases in this key segment.

In [ ]: