Quantium1

December 31, 2023

1 Importing Libraries

[1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from scipy import stats
import plotly.express as px
import plotly.graph_objects as go
```

2 Loading and Exploring Purchase Data (EDA)

[2]:

```
df1 = pd.read_csv("C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL INTERNSHIP/
  ↪purchase_behaviour.csv")

print(df1.head())  # Display the first few rows of the DataFrame

print(df1.describe())  # Display basic statistics of the data

print(df1.info())  # Check data types and missing values
```

```
   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
       LYLTY_CARD_NBR
count    7.263700e+04
mean     1.361859e+05
std      8.989293e+04
min      1.000000e+03
25%      6.620200e+04
50%      1.340400e+05
75%      2.033750e+05
```

```
max      2.373711e+06
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72637 entries, 0 to 72636
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   LYLTY_CARD_NBR    72637 non-null  int64 
 1   LIFESTAGE         72637 non-null  object
 2   PREMIUM_CUSTOMER  72637 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.7+ MB
None
```

2.1 Examining the values of LIFESTAGE

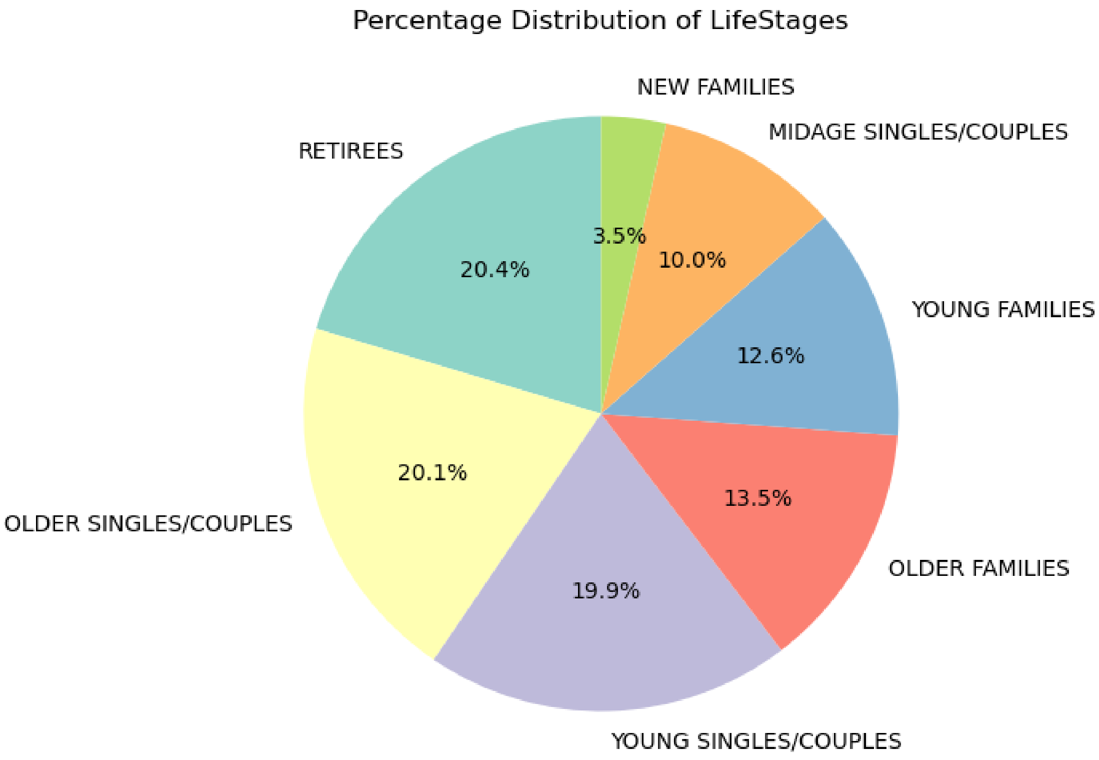
[3]:

```python
customer_data = df1['LIFESTAGE'].value_counts()
print (customer_data)
```

```
RETIREES                  14805
OLDER SINGLES/COUPLES     14609
YOUNG SINGLES/COUPLES     14441
OLDER FAMILIES             9780
YOUNG FAMILIES             9178
MIDAGE SINGLES/COUPLES     7275
NEW FAMILIES               2549
Name: LIFESTAGE, dtype: int64
```

[4]:

```python
# Calculate the percentage distribution of life stages
life_stage_distribution = df1['LIFESTAGE'].value_counts(normalize=True) * 100

# Plotting the pie chart
plt.figure(figsize=(6, 8))
plt.pie(life_stage_distribution, labels=life_stage_distribution.index,␣
 ↪autopct='%1.1f%%', startangle=90, colors=plt.cm.Set3.colors)
plt.title('Percentage Distribution of LifeStages')
plt.show()
```

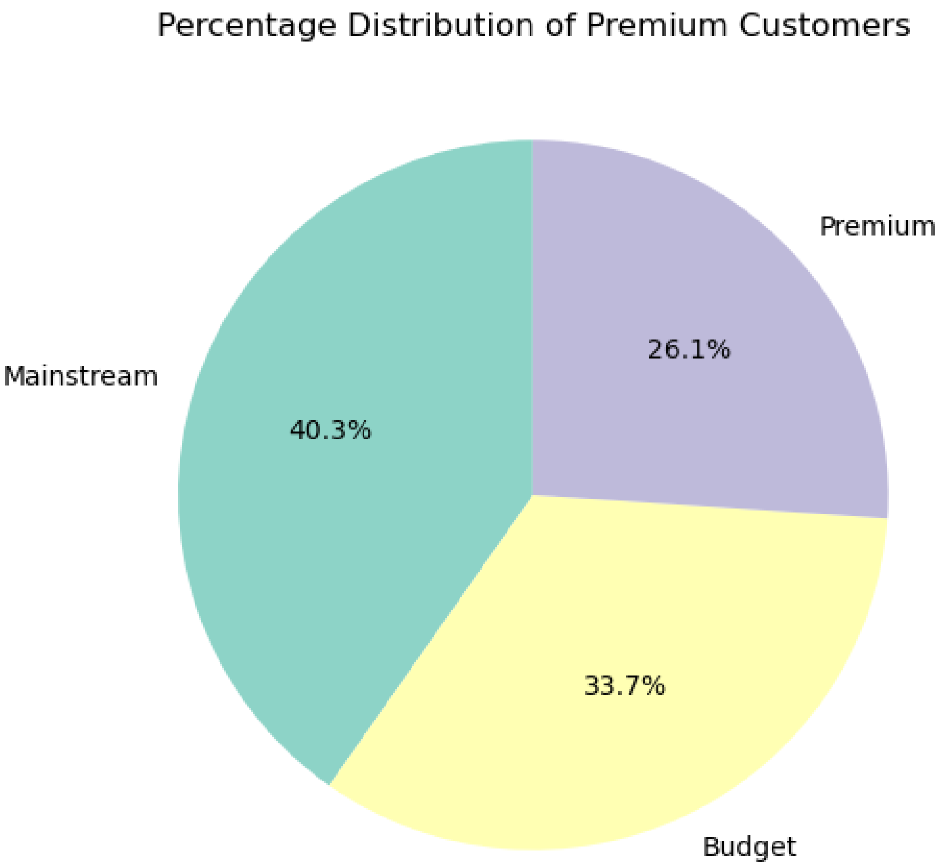


2.2 Examining the values of PREMIUM_CUSTOMER

[5]:
```
customer_data = df1['PREMIUM_CUSTOMER'].value_counts()
print(customer_data)
```

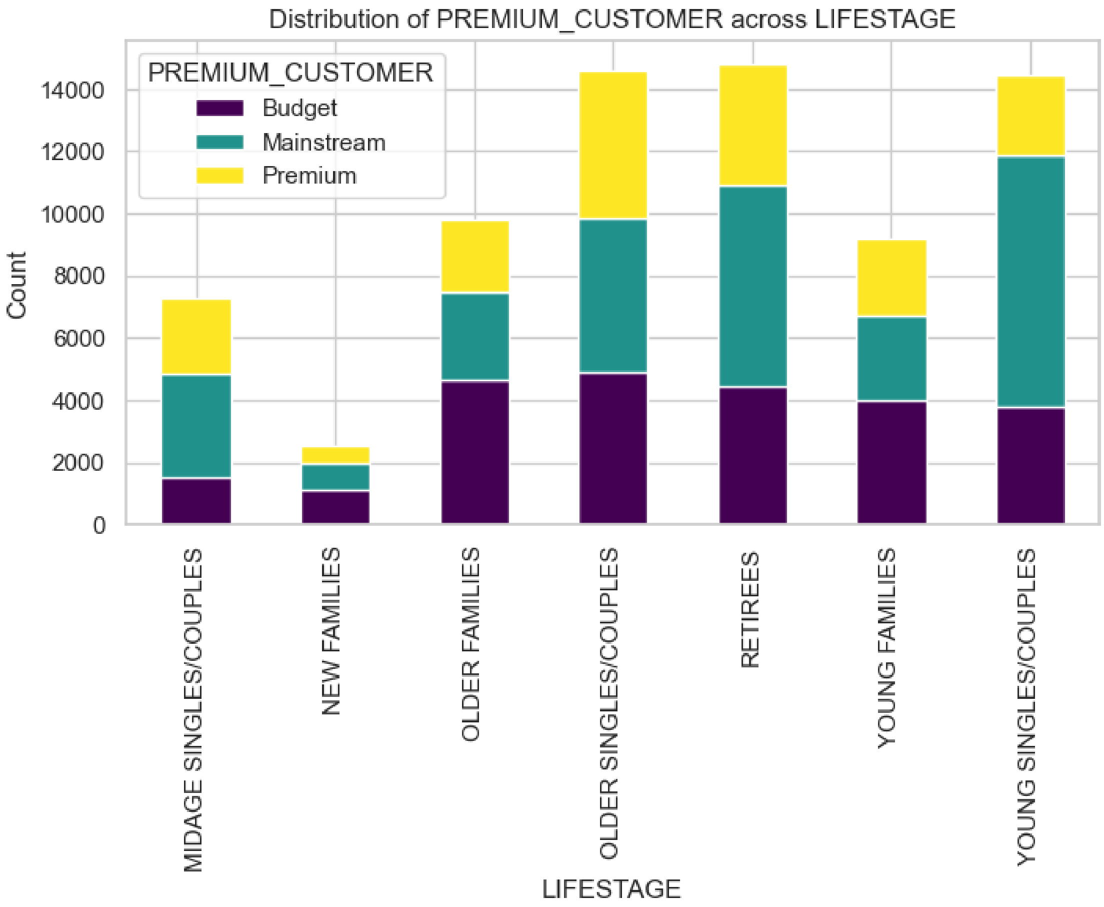
```
Mainstream    29245
Budget        24470
Premium       18922
Name: PREMIUM_CUSTOMER, dtype: int64
```

[6]:
```
# Calculate the percentage distribution of life stages
life_stage_distribution = df1['PREMIUM_CUSTOMER'].value_counts(normalize=True)␣
  ↪* 100

# Plotting the pie chart
plt.figure(figsize=(6, 6))
plt.pie(life_stage_distribution, labels=life_stage_distribution.index,␣
  ↪autopct='%1.1f%%', startangle=90, colors=plt.cm.Set3.colors)
plt.title('Percentage Distribution of Premium Customers')
plt.show()
```



[7]:

```python
# Create a cross-tabulation of LIFESTAGE and PREMIUM_CUSTOMER
cross_tab = pd.crosstab(df1['LIFESTAGE'],df1['PREMIUM_CUSTOMER'])

# Plotting the grouped bar chart
plt.figure(figsize=(8, 4))
sns.set(style="whitegrid")  # Optional styling
cross_tab.plot(kind='bar', stacked=True, colormap="viridis", ax=plt.gca())
plt.title('Distribution of PREMIUM_CUSTOMER across LIFESTAGE')
plt.xlabel('LIFESTAGE')
plt.ylabel('Count')
plt.legend(title='PREMIUM_CUSTOMER')

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



3 Loading and Exploring Transaction Data

[8]:

```
df2 = pd.read_csv("C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL INTERNSHIP/
 ↪transaction_data.csv")

print(df2.head())  # Display the first few rows of the DataFrame

print(df2.describe())  # Display basic statistics of the data

print(df2.info())  # Check data types and missing values
```

```
    DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0  43390          1            1000       1         5
1  43599          1            1307     348        66
2  43605          1            1343     383        61
3  43329          2            2373     974        69
4  43330          2            2426    1038       108
```

```
                                  PROD_NAME  PROD_QTY  TOT_SALES
0    Natural Chip        Compny SeaSalt175g         2        6.0
1                   CCs Nacho Cheese    175g         3        6.3
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
                DATE     STORE_NBR  LYLTY_CARD_NBR        TXN_ID  \
count  264836.000000  264836.00000    2.648360e+05  2.648360e+05
mean    43464.036260     135.08011    1.355495e+05  1.351583e+05
std       105.389282      76.78418    8.057998e+04  7.813303e+04
min     43282.000000       1.00000    1.000000e+03  1.000000e+00
25%     43373.000000      70.00000    7.002100e+04  6.760150e+04
50%     43464.000000     130.00000    1.303575e+05  1.351375e+05
75%     43555.000000     203.00000    2.030942e+05  2.027012e+05
max     43646.000000     272.00000    2.373711e+06  2.415841e+06

            PROD_NBR       PROD_QTY      TOT_SALES
count  264836.000000  264836.000000  264836.000000
mean       56.583157       1.907309       7.304200
std        32.826638       0.643654       3.083226
min         1.000000       1.000000       1.500000
25%        28.000000       2.000000       5.400000
50%        56.000000       2.000000       7.400000
75%        85.000000       2.000000       9.200000
max       114.000000     200.000000     650.000000
<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
None
```

3.1 Creating Additional Features

```
[9]: df2 = pd.read_csv("C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL INTERNSHIP/
      ↪transaction_data.csv")
```

```
# Convert DATE column to a date format
df2['DATE'] = pd.to_datetime(df2['DATE'], origin='2023-12-30')
print(df2.head())
```

```
                           DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0 2023-12-30 00:00:00.000043390          1            1000       1         5
1 2023-12-30 00:00:00.000043599          1            1307     348        66
2 2023-12-30 00:00:00.000043605          1            1343     383        61
3 2023-12-30 00:00:00.000043329          2            2373     974        69
4 2023-12-30 00:00:00.000043330          2            2426    1038       108

                                  PROD_NAME  PROD_QTY  TOT_SALES
0    Natural Chip        Compny SeaSalt175g         2        6.0
1                  CCs Nacho Cheese    175g         3        6.3
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
```

[10]:

```
# Assuming 'DATE' is in datetime format
transaction_counts_by_date = df2.groupby('DATE').size().
 ↪reset_index(name='Transaction_Count')
print(transaction_counts_by_date)
```

```
                             DATE  Transaction_Count
0   2023-12-30 00:00:00.000043282                724
1   2023-12-30 00:00:00.000043283                711
2   2023-12-30 00:00:00.000043284                722
3   2023-12-30 00:00:00.000043285                714
4   2023-12-30 00:00:00.000043286                712
..                            …                …
359 2023-12-30 00:00:00.000043642                723
360 2023-12-30 00:00:00.000043643                709
361 2023-12-30 00:00:00.000043644                730
362 2023-12-30 00:00:00.000043645                745
363 2023-12-30 00:00:00.000043646                744

[364 rows x 2 columns]
```

[11]:

```
# Assuming 'PROD_NAME' is the column containing brand name, product name, and␣
 ↪pack size
df2['brand_name'] = df2['PROD_NAME'].str.extract(r'([a-zA-Z]+)')
df2['product_name'] = df2['PROD_NAME'].str.extract(r'([a-zA-Z\s]+)')
df2['pack_size'] = df2['PROD_NAME'].str.extract(r'(\d+)')

# Convert the extracted pack size to float (if needed)
df2['pack_size'] = df2['pack_size'].astype(float)

# Display the updated DataFrame
```

```
print(df2.head())
```

```
                           DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0 2023-12-30 00:00:00.000043390          1            1000       1         5
1 2023-12-30 00:00:00.000043599          1            1307     348        66
2 2023-12-30 00:00:00.000043605          1            1343     383        61
3 2023-12-30 00:00:00.000043329          2            2373     974        69
4 2023-12-30 00:00:00.000043330          2            2426    1038       108

                                  PROD_NAME  PROD_QTY  TOT_SALES brand_name  \
0    Natural Chip        Compny SeaSalt175g         2        6.0    Natural
1                  CCs Nacho Cheese    175g         3        6.3        CCs
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9     Smiths
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0     Smiths
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8     Kettle

                         product_name  pack_size
0  Natural Chip        Compny SeaSalt      175.0
1                CCs Nacho Cheese          175.0
2  Smiths Crinkle Cut  Chips Chicken       170.0
3               Smiths Chip Thinly  S      175.0
4             Kettle Tortilla ChpsHny      150.0
```

[12]:

```
# Assuming df is your DataFrame
prod_name_counts = df2['PROD_NAME'].value_counts()
print(prod_name_counts)
```

```
Kettle Mozzarella   Basil & Pesto 175g      3304
Kettle Tortilla ChpsHny&Jlpno Chili 150g    3296
Cobs Popd Swt/Chlli &Sr/Cream Chips 110g    3269
Tyrrells Crisps     Ched & Chives 165g      3268
Cobs Popd Sea Salt  Chips 110g              3265
                                            …
RRD Pc Sea Salt     165g                    1431
Woolworths Medium   Salsa 300g              1430
NCC Sour Cream &    Garden Chives 175g      1419
French Fries Potato Chips 175g              1418
WW Crinkle Cut      Original 175g           1410
Name: PROD_NAME, Length: 114, dtype: int64
```

[13]:

```
# Examine the words in PROD_NAME
product_words = df2['PROD_NAME'].str.split(expand=True).stack().
  ↪reset_index(drop=True)
product_words = pd.DataFrame({'words': product_words})

# Removing digits
product_words = product_words[~product_words['words'].str.contains('\d')]
```

```
# Removing special characters
product_words = product_words[product_words['words'].str.isalpha()]

# Look at the most common words
word_counts = product_words['words'].value_counts().reset_index()
word_counts.columns = ['words', 'TOT_SALES']
word_counts = word_counts.sort_values(by='TOT_SALES', ascending=False)
print(word_counts)
```

```
       words  TOT_SALES
0      Chips      49770
1     Kettle      41288
2     Smiths      28860
3       Salt      27976
4     Cheese      27890
..       …          …
163  Whlegrn       1432
164       Pc       1431
165      NCC       1419
166   Garden       1419
167    Fries       1418

[168 rows x 2 columns]
```

[14]:

```
# Note: In Python, the equivalent of grepl is str.contains
# Remove salsa products
df2['SALSA'] = df2['PROD_NAME'].str.contains('salsa', case=False)
df2 = df2[~df2['SALSA']]

# Summarise the data to check for nulls and possible outliers
summary_stats = df2.describe(include='all').transpose()
print(summary_stats)
```

```
                  count unique                                      top  \
DATE             246742    364          2023-12-30 00:00:00.000043458
STORE_NBR      246742.0    NaN                                      NaN
LYLTY_CARD_NBR 246742.0    NaN                                      NaN
TXN_ID         246742.0    NaN                                      NaN
PROD_NBR       246742.0    NaN                                      NaN
PROD_NAME        246742    105  Kettle Mozzarella   Basil & Pesto 175g
PROD_QTY       246742.0    NaN                                      NaN
TOT_SALES      246742.0    NaN                                      NaN
brand_name       246742     28                                   Kettle
product_name     246742    105             Kettle Mozzarella   Basil
pack_size      246742.0    NaN                                      NaN
SALSA            246742      1                                    False

                  freq                          first  \
DATE               865 2023-12-30 00:00:00.000043282
STORE_NBR          NaN                           NaT
LYLTY_CARD_NBR     NaN                           NaT
TXN_ID             NaN                           NaT
PROD_NBR           NaN                           NaT
PROD_NAME         3304                           NaT
PROD_QTY           NaN                           NaT
TOT_SALES          NaN                           NaT
brand_name       41288                           NaT
product_name      3304                           NaT
pack_size          NaN                           NaT
SALSA           246742                           NaT

                                        last           mean           std  \
DATE           2023-12-30 00:00:00.000043646            NaN           NaN
STORE_NBR                                NaT     135.051098     76.787096
LYLTY_CARD_NBR                           NaT  135530.984956  80715.280765
TXN_ID                                   NaT  135131.098848  78147.717692
PROD_NBR                                 NaT      56.351789     33.695428
PROD_NAME                                NaT            NaN           NaN
PROD_QTY                                 NaT       1.908062      0.659831
TOT_SALES                                NaT       7.321322      3.077828
brand_name                               NaT            NaN           NaN
product_name                             NaT            NaN           NaN
pack_size                                NaT     175.585178     59.434727
SALSA                                    NaT            NaN           NaN

                   min       25%       50%        75%        max
DATE               NaN       NaN       NaN        NaN        NaN
STORE_NBR          1.0      70.0     130.0      203.0      272.0
LYLTY_CARD_NBR  1000.0   70015.0  130367.0   203084.0  2373711.0
TXN_ID             1.0  67569.25  135183.0  202653.75  2415841.0
PROD_NBR           1.0      26.0      53.0       87.0      114.0
PROD_NAME          NaN       NaN       NaN        NaN        NaN
PROD_QTY           1.0       2.0       2.0        2.0      200.0
TOT_SALES          1.7       5.8       7.4        8.8      650.0
brand_name         NaN       NaN       NaN        NaN        NaN
product_name       NaN       NaN       NaN        NaN        NaN
pack_size         70.0     150.0     170.0      175.0      380.0
SALSA              NaN       NaN       NaN        NaN        NaN

C:\Users\Asus\AppData\Local\Temp\ipykernel_13204\2321965016.py:7: FutureWarning:
Treating datetime data as categorical rather than numeric in `.describe` is
deprecated and will be removed in a future version of pandas. Specify
`datetime_is_numeric=True` to silence this warning and adopt the future behavior
now.
  summary_stats = df2.describe(include='all').transpose()
```

3.2 Examining Pack Size

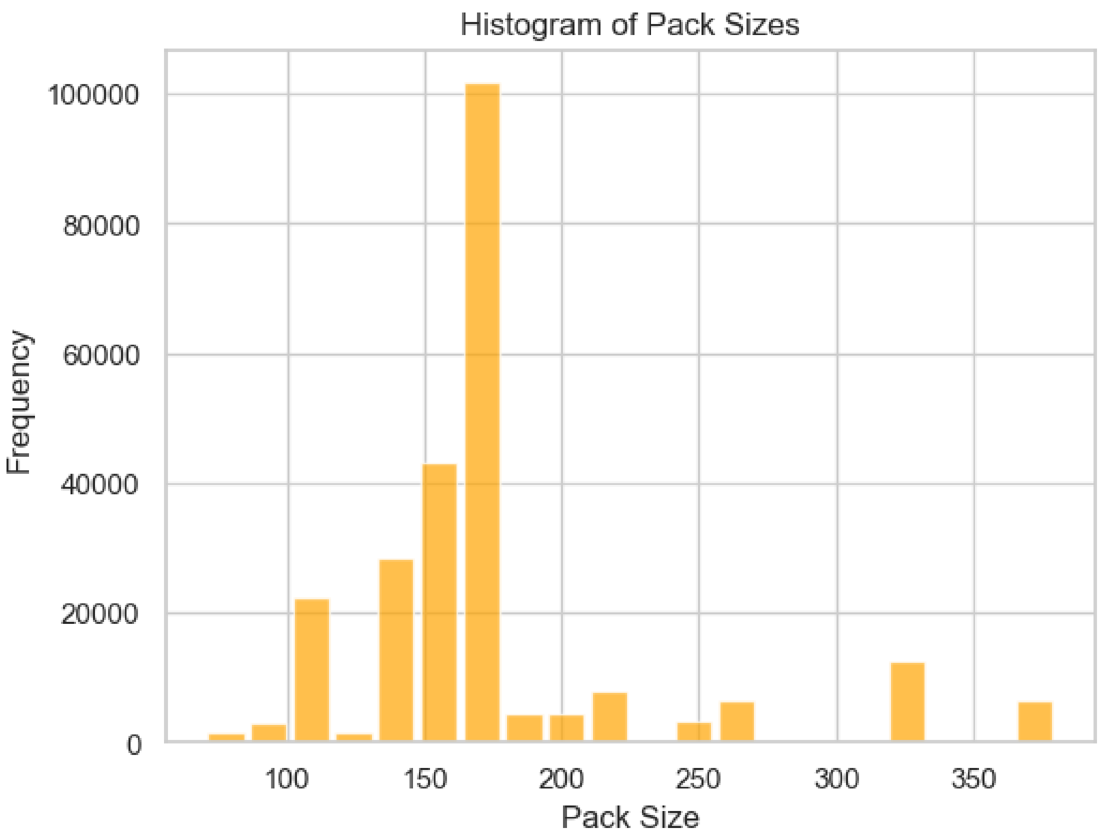
[15]:

```
# Assuming 'PROD_NAME' is the column containing product names
# Extract digits from 'PROD_NAME' to get pack size
df2['PACK_SIZE'] = df2['PROD_NAME'].str.extract('(\d+)')

# Convert 'PACK_SIZE' to numeric
df2['PACK_SIZE'] = pd.to_numeric(df2['PACK_SIZE'])

# Check the pack sizes
pack_size_counts = df2['PACK_SIZE'].value_counts().sort_index()
print(pack_size_counts)
```

```
70      1507
90      3008
110    22387
125     1454
134    25102
135     3257
150    40203
160     2970
165    15297
170    19983
175    66390
180     1468
190     2995
200     4473
210     6272
220     1564
250     3169
270     6285
330    12540
380     6418
Name: PACK_SIZE, dtype: int64
```

[16]:

```
# Assuming 'PACK_SIZE' is the column containing pack sizes
plt.hist(df2['PACK_SIZE'], bins=20, color='orange', alpha=0.7, rwidth=0.85)
plt.title('Histogram of Pack Sizes')
plt.xlabel('Pack Size')
plt.ylabel('Frequency')
plt.show()
```



3.3 Examining Brands and their Total Sales

[17]:

```python
# Assuming 'PROD_NAME' is the column containing product names
df2['BRAND'] = df2['PROD_NAME'].apply(lambda x: x[:x.find(' ')])

# Checking brands
brand_counts = df2['BRAND'].value_counts().reset_index()
brand_counts.columns = ['BRAND', 'TOT_SALES']
brand_counts = brand_counts.sort_values(by='TOT_SALES', ascending=False)
print(brand_counts)
```

```
         BRAND  TOT_SALES
0       Kettle      41288
1       Smiths      27390
2     Pringles      25102
3      Doritos      22041
4        Thins      14075
5          RRD      11894
6     Infuzions     11057
7           WW      10320
8         Cobs       9693
```

```
9     Tostitos       9471
10    Twisties       9454
11    Tyrrells       6442
12       Grain       6272
13     Natural       6050
14    Cheezels       4603
15         CCs       4551
16         Red       4427
17      Dorito       3185
18      Infzns       3144
19       Smith       2963
20     Cheetos       2927
21       Snbts       1576
22      Burger       1564
23  Woolworths       1516
24     GrnWves       1468
25    Sunbites       1432
26         NCC       1419
27      French       1418
```

[18]:

```python
# Clean brand names
brand_mapping = {
    "RED": "RRD",
    "SNBTS": "SUNBITES",
    "INFZNS": "INFUZIONS",
    "WW": "WOOLWORTHS",
    "SMITH": "SMITHS",
    "NCC": "NATURAL",
    "DORITO": "DORITOS",
    "GRAIN": "GRNWVES"
}

df2['BRAND'] = df2['BRAND'].replace(brand_mapping)

# Check again
brand_counts = df2['BRAND'].value_counts().reset_index()
brand_counts.columns = ['BRAND', 'TOT_SALES']
brand_counts = brand_counts.sort_values(by='TOT_SALES')
print(brand_counts)
```

```
         BRAND  TOT_SALES
27      French       1418
26     NATURAL       1419
25    Sunbites       1432
24     GrnWves       1468
23  Woolworths       1516
22      Burger       1564
21       Snbts       1576
```

```
20     Cheetos       2927
19       Smith       2963
18      Infzns       3144
17      Dorito       3185
16         Red       4427
15         CCs       4551
14    Cheezels       4603
13     Natural       6050
12       Grain       6272
11    Tyrrells       6442
10    Twisties       9454
9     Tostitos       9471
8         Cobs       9693
7   WOOLWORTHS      10320
6    Infuzions      11057
5          RRD      11894
4        Thins      14075
3      Doritos      22041
2     Pringles      25102
1       Smiths      27390
0       Kettle      41288
```

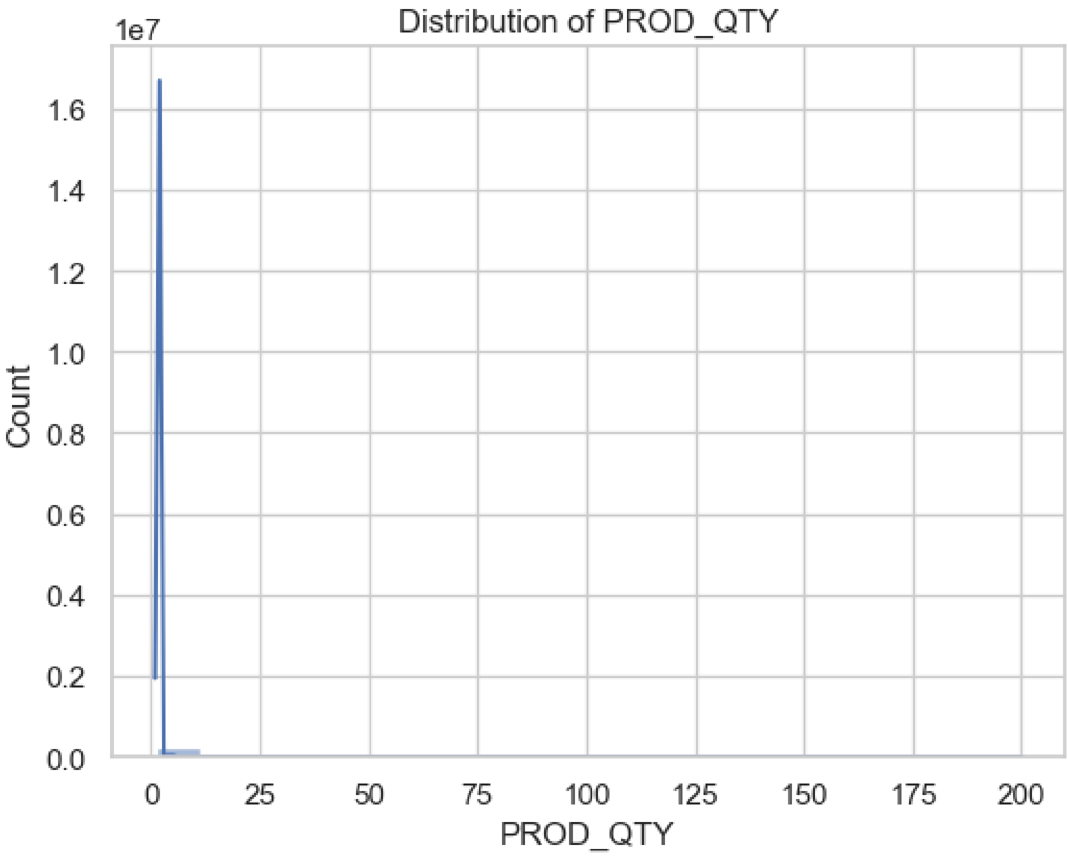
4 Identifying and Handling Outliers

[19]:

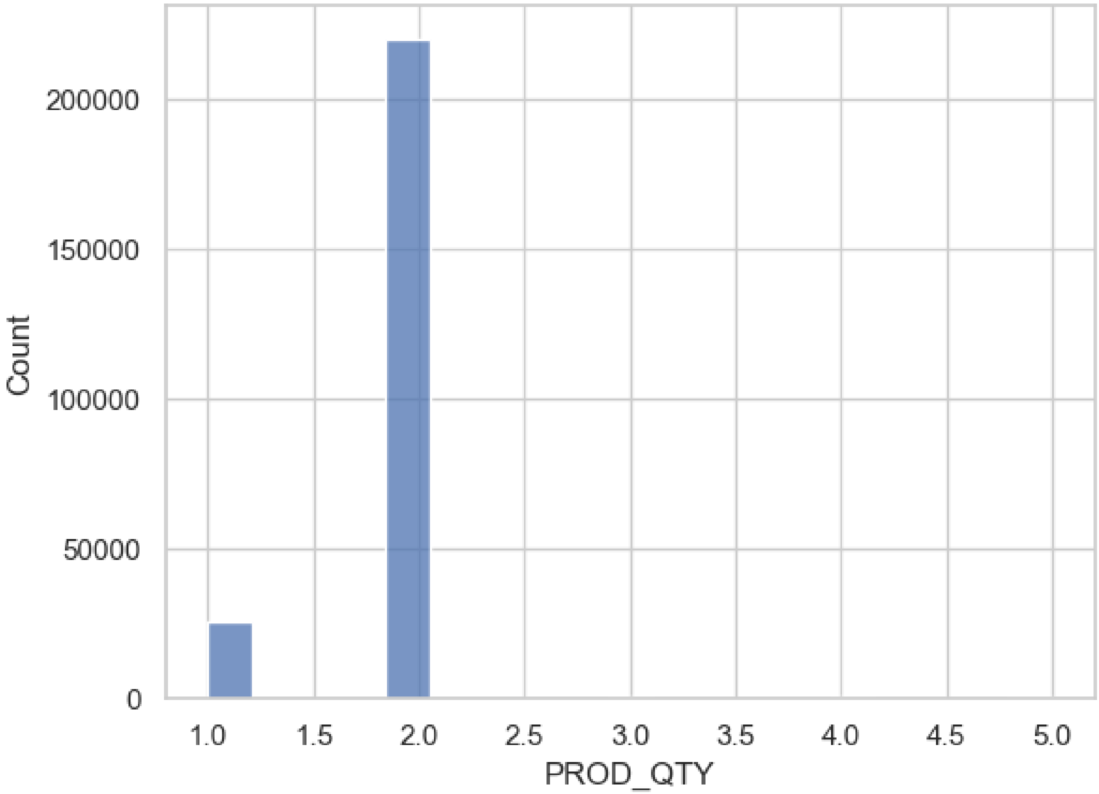
```
# Filter the dataset to find the outlier
outlier_data = df2[df2['PROD_QTY'] == 200]
print(outlier_data)

## Visualize the distribution of the variable of interest (e.g., 'spend')
sns.histplot(df2['PROD_QTY'], kde=True)
plt.title('Distribution of PROD_QTY')
plt.show()

# Calculate Z-scores
z_scores = np.abs(stats.zscore(df2['PROD_QTY']))

# Define a threshold for identifying outliers (e.g., Z-score > 3)
outliers = (z_scores > 200)

# Identify and print the outliers
outlier_values = df2['PROD_QTY'][outliers]
print("Outlier values:")
print(outlier_values)

# Remove outliers from the dataset
df_no_outliers = df2[~outliers]
```

```
# Display the box plot after removing outliers for comparison
sns.histplot(x=df_no_outliers['PROD_QTY'])
plt.show()
```

```
                               DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  \
69762 2023-12-30 00:00:00.000043331        226          226000  226201
69763 2023-12-30 00:00:00.000043605        226          226000  226210

       PROD_NBR                         PROD_NAME  PROD_QTY  TOT_SALES  \
69762         4  Dorito Corn Chp     Supreme 380g       200      650.0
69763         4  Dorito Corn Chp     Supreme 380g       200      650.0

      brand_name                  product_name  pack_size  SALSA  PACK_SIZE  \
69762     Dorito  Dorito Corn Chp     Supreme       380.0  False        380
69763     Dorito  Dorito Corn Chp     Supreme       380.0  False        380

        BRAND
69762  Dorito
69763  Dorito
```



```
Outlier values:
69762    200
69763    200
Name: PROD_QTY, dtype: int64
```

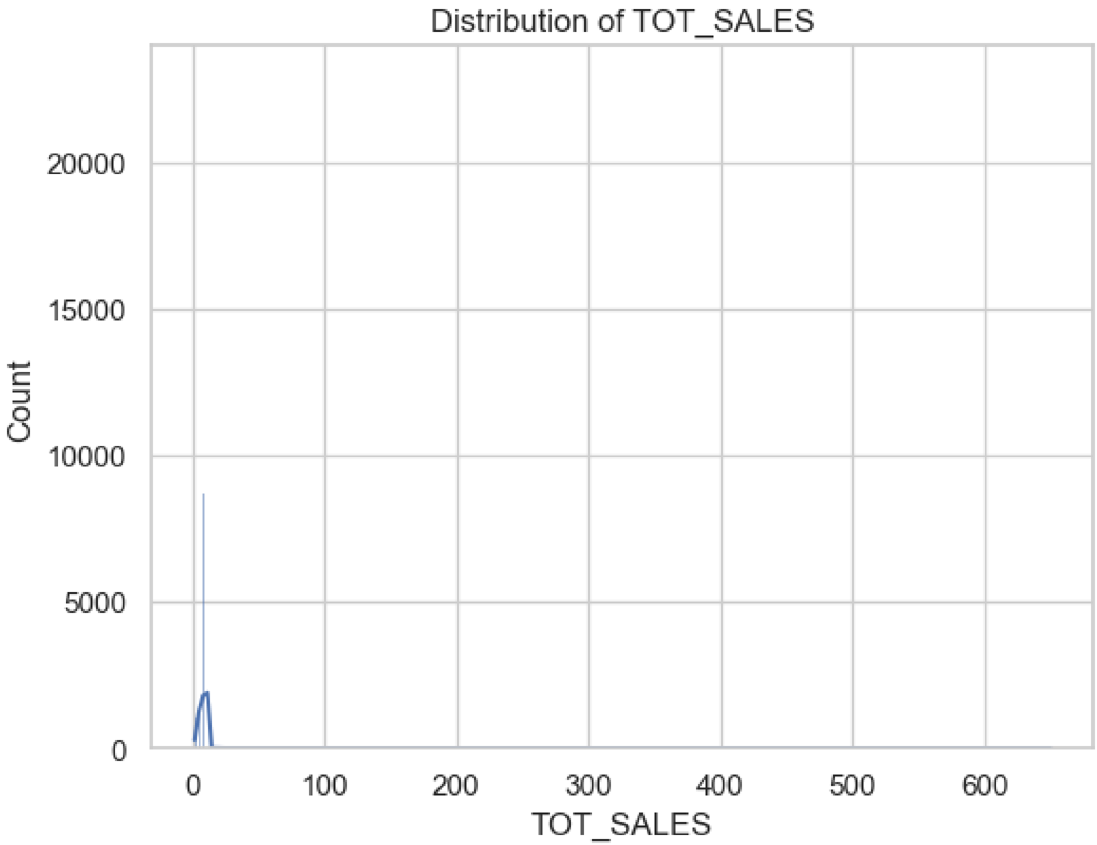


[20]:

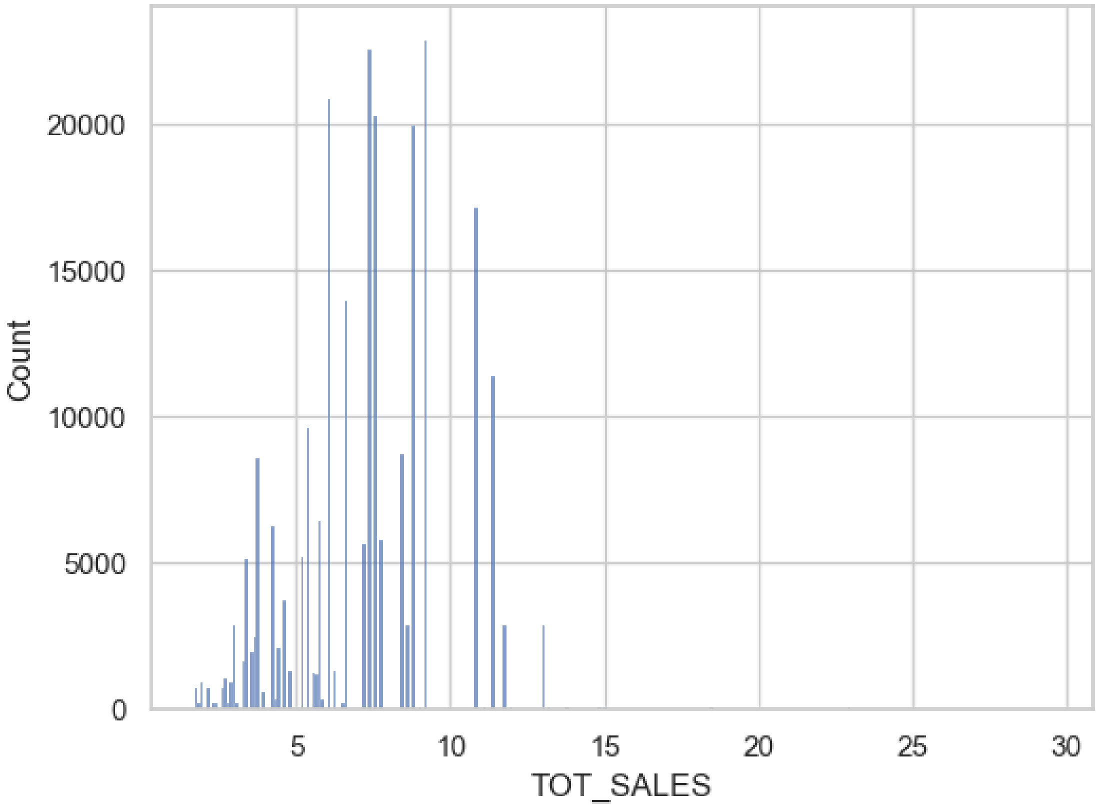
```
# Filter the dataset to find the outlier
outlier_data = df2[df2['TOT_SALES'] == 200]
print(outlier_data)

## Visualize the distribution of the variable of interest (e.g., 'spend')
sns.histplot(df2['TOT_SALES'], kde=True)
plt.title('Distribution of TOT_SALES')
plt.show()

# Calculate Z-scores
z_scores = np.abs(stats.zscore(df2['TOT_SALES']))

# Define a threshold for identifying outliers (e.g., Z-score > 3)
outliers = (z_scores > 100)

# Identify and print the outliers
```

```python
outlier_values = df2['TOT_SALES'][outliers]
print("Outlier values:")
print(outlier_values)

# Remove outliers from the dataset
df_no_outliers = df2[~outliers]

# Display the box plot after removing outliers for comparison
sns.histplot(x=df_no_outliers['TOT_SALES'])
plt.show()
```

```
Empty DataFrame
Columns: [DATE, STORE_NBR, LYLTY_CARD_NBR, TXN_ID, PROD_NBR, PROD_NAME,
PROD_QTY, TOT_SALES, brand_name, product_name, pack_size, SALSA, PACK_SIZE,
BRAND]
Index: []
```



```
Outlier values:
69762    650.0
69763    650.0
Name: TOT_SALES, dtype: float64
```



5 Combining Purchase and Transaction Dataset

[21]:
```
import pandas as pd

# Assuming df_purchase and df_transaction are your datasets
# Replace 'LYLTY_CARD_NBR' with the actual common column name
merged_data = pd.merge(df1, df2, on='LYLTY_CARD_NBR', how='inner')

# Now merged_data contains columns from both datasets based on the common␣
↪'LYLTY_CARD_NBR'
print(merged_data)
```

```
        LYLTY_CARD_NBR              LIFESTAGE PREMIUM_CUSTOMER  \
0                 1000  YOUNG SINGLES/COUPLES          Premium
1                 1002  YOUNG SINGLES/COUPLES       Mainstream
2                 1003         YOUNG FAMILIES           Budget
3                 1003         YOUNG FAMILIES           Budget
4                 1004  OLDER SINGLES/COUPLES       Mainstream
…                  …                    …              …
246737         2370651 MIDAGE SINGLES/COUPLES       Mainstream
246738         2370701         YOUNG FAMILIES       Mainstream
```

```
246739         2370751          YOUNG FAMILIES          Premium
246740         2370961          OLDER FAMILIES           Budget
246741         2373711   YOUNG SINGLES/COUPLES       Mainstream

                                DATE  STORE_NBR  TXN_ID  PROD_NBR  \
0      2023-12-30 00:00:00.000043390          1       1         5
1      2023-12-30 00:00:00.000043359          1       2        58
2      2023-12-30 00:00:00.000043531          1       3        52
3      2023-12-30 00:00:00.000043532          1       4       106
4      2023-12-30 00:00:00.000043406          1       5        96
…                               …       …     …       …
246737 2023-12-30 00:00:00.000043315         88  240350         4
246738 2023-12-30 00:00:00.000043442         88  240378        24
246739 2023-12-30 00:00:00.000043374         88  240394        60
246740 2023-12-30 00:00:00.000043397         88  240480        70
246741 2023-12-30 00:00:00.000043448         88  241815        16

                                       PROD_NAME  PROD_QTY  TOT_SALES  \
0         Natural Chip        Compny SeaSalt175g         2        6.0
1          Red Rock Deli Chikn&Garlic Aioli 150g         1        2.7
2          Grain Waves Sour    Cream&Chives 210G         1        3.6
3         Natural ChipCo      Hony Soy Chckn175g         1        3.0
4                 WW Original Stacked Chips 160g         1        1.9
…                                           …      …        …
246737          Dorito Corn Chp     Supreme 380g         2       13.0
246738     Grain Waves         Sweet Chilli 210g         2        7.2
246739      Kettle Tortilla ChpsFeta&Garlic 150g         2        9.2
246740   Tyrrells Crisps     Lightly Salted 165g         2        8.4
246741  Smiths Crinkle Chips Salt & Vinegar 330g         2       11.4

       brand_name                         product_name  pack_size  SALSA  \
0         Natural   Natural Chip        Compny SeaSalt      175.0  False
1             Red                  Red Rock Deli Chikn      150.0  False
2           Grain            Grain Waves Sour    Cream      210.0  False
3         Natural   Natural ChipCo      Hony Soy Chckn      175.0  False
4              WW           WW Original Stacked Chips       160.0  False
…           …                                  …        …    …
246737     Dorito         Dorito Corn Chp     Supreme       380.0  False
246738      Grain    Grain Waves         Sweet Chilli       210.0  False
246739     Kettle             Kettle Tortilla ChpsFeta      150.0  False
246740   Tyrrells  Tyrrells Crisps     Lightly Salted       165.0  False
246741     Smiths           Smiths Crinkle Chips Salt       330.0  False

        PACK_SIZE       BRAND
0             175     Natural
1             150         Red
2             210       Grain
3             175     Natural
```

```
4              160  WOOLWORTHS
…           …        …
246737         380      Dorito
246738         210       Grain
246739         150      Kettle
246740         165    Tyrrells
246741         330      Smiths

[246742 rows x 16 columns]
```

6 Save to Excel

[22]:

```python
# Assuming your merged DataFrame is named merged_data
null_lifestage_count = merged_data['LIFESTAGE'].isnull().sum()
null_premium_customer_count = merged_data['PREMIUM_CUSTOMER'].isnull().sum()

print("Number of null values in LIFESTAGE:", null_lifestage_count)
print("Number of null values in PREMIUM_CUSTOMER:", null_premium_customer_count)
```

```
Number of null values in LIFESTAGE: 0
Number of null values in PREMIUM_CUSTOMER: 0
```

[23]:

```python
merged_data.to_excel("C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL␣
 ↪INTERNSHIP/new_data.xlsx", index=False)
print("\nData saved to 'C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL␣
 ↪INTERNSHIP/new_data.xlsx'")
```

```

Data saved to 'C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL
INTERNSHIP/new_data.xlsx'
```

7 Define Metrics

[24]:

```python
df3 = pd.read_excel("C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL␣
 ↪INTERNSHIP/new_data.xlsx")

# Total Spend per Customer
total_spend_per_customer = df3.groupby('LYLTY_CARD_NBR')['TOT_SALES'].sum()
df3['total_spend_per_customer'] = total_spend_per_customer
print("Total Spend per Customer:")
print(total_spend_per_customer)
print()

# Average Spend per Customer
average_spend_per_customer = df3.groupby('LYLTY_CARD_NBR')['TOT_SALES'].mean()
df3['average_spend_per_customer'] = average_spend_per_customer
print("Average Spend per Customer:")
print(average_spend_per_customer)
print()

# Frequency of Purchase
purchase_frequency = df3.groupby('LYLTY_CARD_NBR').size()
df3['purchase_frequency'] = purchase_frequency
print("Frequency of Purchase:")
print(purchase_frequency)
print()

# Average Pack Size per Customer
average_pack_size_per_customer = df3.groupby('LYLTY_CARD_NBR')['pack_size'].
  ↪mean()
df3['average_pack_size_per_customer'] = average_pack_size_per_customer
print("Average Pack Size per Customer:")
print(average_pack_size_per_customer)
print()

# Save to Excel
result_df = pd.DataFrame({
    'Total Spend per Customer': total_spend_per_customer,
    'Average Spend per Customer': average_spend_per_customer,
    'Frequency of Purchase': purchase_frequency,
    'Average Pack Size per Customer': average_pack_size_per_customer,
})

result_df.to_excel("C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL INTERNSHIP/
  ↪metrics_data.xlsx", index=False)
print("\nData saved to 'C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL␣
  ↪INTERNSHIP/metrics_data.xlsx'")
```

```
Total Spend per Customer:
LYLTY_CARD_NBR
1000        6.0
1002        2.7
1003        6.6
1004        1.9
1005        2.8
           …
2370651    13.0
2370701     7.2
2370751     9.2
2370961     8.4
2373711    11.4
Name: TOT_SALES, Length: 71288, dtype: float64

Average Spend per Customer:
```

```
LYLTY_CARD_NBR
1000        6.0
1002        2.7
1003        3.3
1004        1.9
1005        2.8
           …
2370651    13.0
2370701     7.2
2370751     9.2
2370961     8.4
2373711    11.4
Name: TOT_SALES, Length: 71288, dtype: float64

Frequency of Purchase:
LYLTY_CARD_NBR
1000       1
1002       1
1003       2
1004       1
1005       1
          ..
2370651    1
2370701    1
2370751    1
2370961    1
2373711    1
Length: 71288, dtype: int64

Average Pack Size per Customer:
LYLTY_CARD_NBR
1000       175.0
1002       150.0
1003       192.5
1004       160.0
1005       165.0
           …
2370651    380.0
2370701    210.0
2370751    150.0
2370961    165.0
2373711    330.0
Name: pack_size, Length: 71288, dtype: float64


Data saved to 'C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL
INTERNSHIP/metrics_data.xlsx'
```

8 Customer Segmentation

[25]:

```
df3 = pd.read_excel("C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL␣
  ↪INTERNSHIP/new_data.xlsx")

# Brand Metrics
brand_metrics = df3.groupby('brand_name')['TOT_SALES'].agg(['mean', 'median']).
  ↪reset_index()
brand_metrics.columns = ['Brand', 'Mean Sales by Brand', 'Median Sales by␣
  ↪Brand']
print("Brand Metrics:")
print(brand_metrics)
print()

# Product Metrics
product_metrics = df3.groupby('product_name')['TOT_SALES'].agg(['mean',␣
  ↪'median']).reset_index()
product_metrics.columns = ['Product', 'Mean Sales by Product', 'Median Sales by␣
  ↪Product']
print("Product Metrics:")
print(product_metrics)
print()

# Save to Excel
result_df = pd.merge(brand_metrics, product_metrics, how='outer',␣
  ↪left_on='Brand', right_on='Product').fillna('')

result_df.to_excel("C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL INTERNSHIP/
  ↪brand_product_metrics.xlsx", index=False)
print("\nData saved to 'C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL␣
  ↪INTERNSHIP/brand_product_metrics.xlsx'")
```

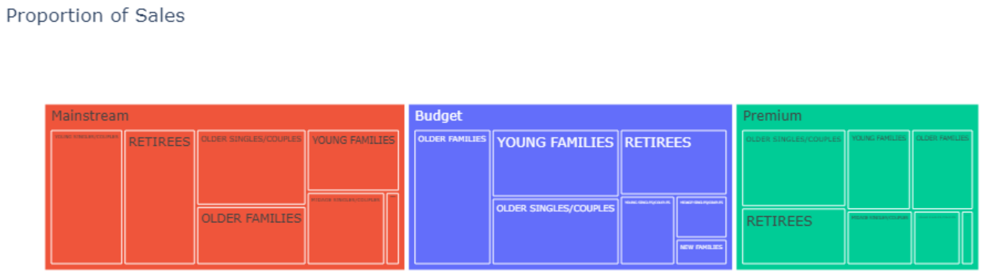
```
Brand Metrics:
        Brand  Mean Sales by Brand  Median Sales by Brand
0      Burger             4.367647                    4.6
1         CCs             3.972512                    4.2
2     Cheetos             5.768534                    5.6
3    Cheezels             8.696481                   11.4
4        Cobs             7.280491                    7.6
5      Dorito            12.669388                   13.0
6     Doritos             8.496797                    8.8
7      French             5.591678                    6.0
8       Grain             6.863648                    7.2
9     GrnWves             5.836785                    6.2
10   Infuzions            6.895867                    7.6
11     Infzns             7.251908                    7.6
12     Kettle             9.451652                    9.2
```

```
13         NCC             5.670190                    6.0
14     Natural             5.664793                    6.0
15    Pringles             7.077344                    7.4
16         RRD             5.461115                    6.0
17         Red             5.117009                    5.4
18       Smith             4.921836                    5.2
19      Smiths             7.408127                    6.0
20       Snbts             3.220939                    3.4
21    Sunbites             3.212430                    3.4
22       Thins             6.312789                    6.6
23    Tostitos             8.424623                    8.8
24    Twisties             8.623027                    9.2
25    Tyrrells             8.017293                    8.4
26          WW             3.477665                    3.8
27  Woolworths             3.410026                    3.6

Product Metrics:
                             Product  Mean Sales by Product  \
0                       Burger Rings                4.367647
1                CCs Nacho Cheese                   3.979907
2                       CCs Original                3.994716
3                CCs Tasty Cheese                   3.943470
4                        Cheetos Chs                6.249696
..                               …                     …
100        WW Original Corn    Chips                3.590301
101        WW Original Stacked Chips                3.580229
102                   WW Sour Cream                 3.589885
103  WW Supreme Cheese   Corn Chips                 3.572101
104        Woolworths Cheese   Rings                3.410026

     Median Sales by Product
0                        4.6
1                        4.2
2                        4.2
3                        4.2
4                        6.6
..                       …
100                      3.8
101                      3.8
102                      3.8
103                      3.8
104                      3.6

[105 rows x 3 columns]


Data saved to 'C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL
INTERNSHIP/brand_product_metrics.xlsx'
```

8.1 Total sales by LIFESTAGE and PREMIUM_CUSTOMER

[26]:

```
df3 = pd.read_excel("C:/Users/Asus/Desktop/Forage/QUANTIUM DA VIRTUAL␣
 ↪INTERNSHIP/new_data.xlsx")

# Assuming your DataFrame is named 'sales'
sales = df3.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).agg({'TOT_SALES':␣
 ↪'sum'}).reset_index()

# Create a mosaic plot
fig = px.treemap(sales,
                 path=['PREMIUM_CUSTOMER', 'LIFESTAGE'],
                 values='TOT_SALES',
                 color='PREMIUM_CUSTOMER',
                 title='Proportion of Sales',
                 )

# Update layout for better readability
fig.update_layout(
    xaxis=dict(tickangle=-45),
    yaxis=dict(title='Premium Customer Flag'),
    treemapcolorway=['lightblue', 'lightgreen', 'orange'],  # Customize colors
)

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

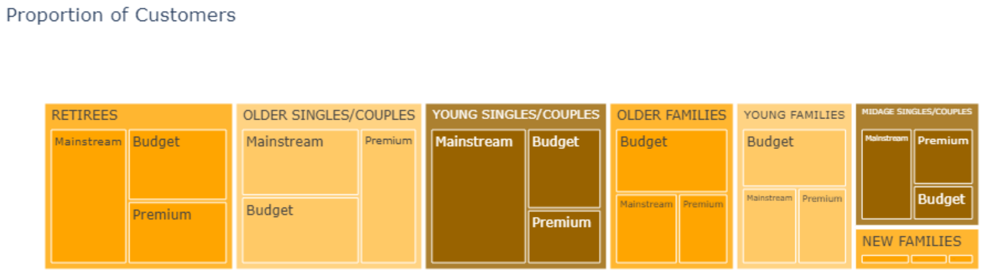


8.2 Number of customers by LIFESTAGE and PREMIUM_CUSTOMER

[27]:

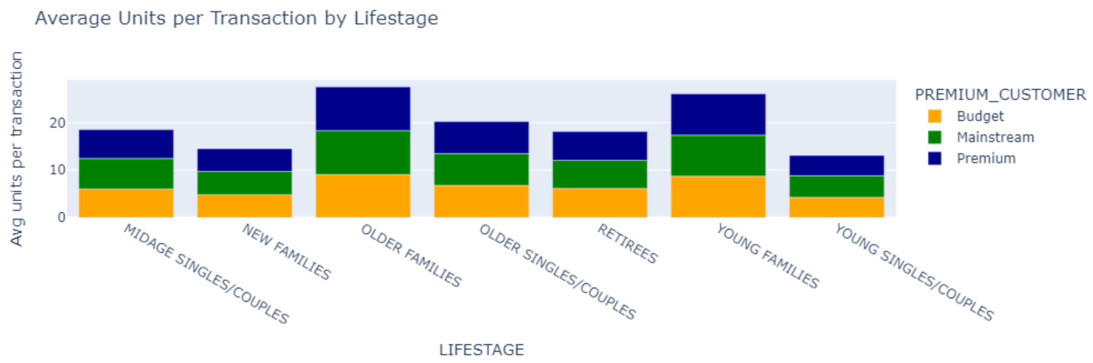
```
customers = df3.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).
 ↪nunique()['LYLTY_CARD_NBR'].reset_index()

# Create an interactive mosaic plot
```

```
fig = px.treemap(customers,
                 path=['LIFESTAGE', 'PREMIUM_CUSTOMER'],
                 values='LYLTY_CARD_NBR',
                 title='Proportion of Customers',
                 labels={'LYLTY_CARD_NBR': 'Number of Customers'},
                 color_discrete_sequence=['orange'])

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



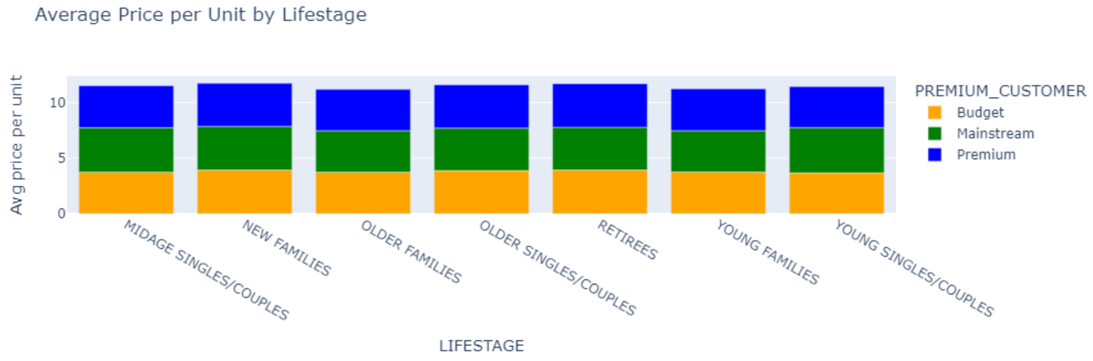
8.3 Average number of units per customer by LIFESTAGE and PREMIUM_CUSTOMER

[28]:
```
avg_units = df3.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).apply(lambda x:␣
↪x['PROD_QTY'].sum() / x['LYLTY_CARD_NBR'].nunique()).reset_index(name='AVG')

# Create an interactive bar chart with different colors for premium customers
fig = px.bar(avg_units,
             x='LIFESTAGE',
             y='AVG',
             color='PREMIUM_CUSTOMER',
             title='Average Units per Transaction by Lifestage',
             labels={'AVG': 'Avg units per transaction'},
             color_discrete_map={'Premium': 'darkblue', 'Mainstream': 'green',␣
↪'Budget': 'orange'})  # Specify colors for each premium customer category

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



8.4 Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER

[29]:
```
avg_price = df3.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).apply(lambda x:␣
↪x['TOT_SALES'].sum() / x['PROD_QTY'].sum()).reset_index(name='AVG')

# Create an interactive bar chart with different colors for premium customers
fig = px.bar(avg_price,
             x='LIFESTAGE',
             y='AVG',
             color='PREMIUM_CUSTOMER',
             title='Average Price per Unit by Lifestage',
             labels={'AVG': 'Avg price per unit'},
             color_discrete_map={'Premium': 'blue', 'Mainstream': 'green',␣
↪'Budget': 'orange'})  # Specify colors for each premium customer category

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



[30]:

```
# Extracting the relevant data for the t-test
mainstream_data = df3[(df3['LIFESTAGE'].isin(["YOUNG SINGLES/COUPLES", "MIDAGE␣
 ↪SINGLES/COUPLES"])) & (df3['PREMIUM_CUSTOMER'] == "Mainstream")]['TOT_SALES']
other_data = df3[(df3['LIFESTAGE'].isin(["YOUNG SINGLES/COUPLES", "MIDAGE␣
 ↪SINGLES/COUPLES"])) & (df3['PREMIUM_CUSTOMER'] != "Mainstream")]['TOT_SALES']

# Perform an independent t-test
t_stat, p_value = stats.ttest_ind(mainstream_data, other_data,␣
 ↪alternative="greater")

# Display the results
print(f'T-statistic: {t_stat}')
print(f'P-value: {p_value}')
```

```
T-statistic: 33.20052175140063
P-value: 9.958402395545195e-240
```

8.5 Deep dive into Mainstream, young singles/couples

[31]:

```
# Create dataframes for the specified segment and other customers
segment1 = df3[(df3['LIFESTAGE'] == "YOUNG SINGLES/COUPLES") &␣
 ↪(df3['PREMIUM_CUSTOMER'] == "Mainstream")]
other = df3[~((df3['LIFESTAGE'] == "YOUNG SINGLES/COUPLES") &␣
 ↪(df3['PREMIUM_CUSTOMER'] == "Mainstream"))]

# Calculate quantities for segment and other customers
quantity_segment1 = segment1['PROD_QTY'].sum()
quantity_other = other['PROD_QTY'].sum()

# Calculate brand proportions for the segment and other customers
quantity_segment1_by_brand = segment1.groupby('BRAND')['PROD_QTY'].sum() /␣
 ↪quantity_segment1
quantity_other_by_brand = other.groupby('BRAND')['PROD_QTY'].sum() /␣
 ↪quantity_other

# Merge dataframes and calculate affinity to brand
brand_proportions = pd.merge(quantity_segment1_by_brand,␣
 ↪quantity_other_by_brand, left_index=True, right_index=True,␣
 ↪suffixes=('_targetSegment', '_other'))
brand_proportions['affinityToBrand'] =␣
 ↪brand_proportions['PROD_QTY_targetSegment'] /␣
 ↪brand_proportions['PROD_QTY_other']

# Display brand proportions sorted by affinity
brand_proportions.sort_values(by='affinityToBrand', ascending=False)
```

```
[31]:             PROD_QTY_targetSegment  PROD_QTY_other  affinityToBrand
      BRAND
      Tyrrells                  0.031553        0.025669         1.229227
      Twisties                  0.046184        0.037842         1.220443
      Doritos                   0.107053        0.088234         1.213293
      Kettle                    0.197985        0.165401         1.196998
      Tostitos                  0.045411        0.037943         1.196815
      Infzns                    0.014934        0.012562         1.188884
      Pringles                  0.119420        0.100542         1.187764
      Grain                     0.029124        0.025098         1.160386
      Dorito                    0.015707        0.013669         1.149162
      Cobs                      0.044638        0.039013         1.144177
      Infuzions                 0.049745        0.044450         1.119104
      Thins                     0.060373        0.056934         1.060399
      Cheezels                  0.017971        0.018630         0.964641
      Smiths                    0.089772        0.112112         0.800737
      French                    0.003948        0.005753         0.686201
      Cheetos                   0.008033        0.012055         0.666346
      RRD                       0.032022        0.049106         0.652107
      Red                       0.011787        0.018326         0.643209
      Natural                   0.015956        0.024958         0.639313
      NATURAL                   0.003644        0.005868         0.620996
      CCs                       0.011180        0.018878         0.592222
      GrnWves                   0.003589        0.006061         0.592083
      Smith                     0.006598        0.012357         0.533923
      Snbts                     0.003478        0.006581         0.528518
      WOOLWORTHS                0.021256        0.043010         0.494212
      Sunbites                  0.002871        0.005987         0.479492
      Woolworths                0.002843        0.006372         0.446241
      Burger                    0.002926        0.006590         0.444005
```

8.6 Preferred pack size compared to the rest of the population

```
[32]: # Calculate pack proportions for the segment and other customers
      quantity_segment1_by_pack = segment1.groupby('PACK_SIZE')['PROD_QTY'].sum() /␣
        ↪quantity_segment1
      quantity_other_by_pack = other.groupby('PACK_SIZE')['PROD_QTY'].sum() /␣
        ↪quantity_other

      # Merge dataframes and calculate affinity to pack
      pack_proportions = pd.merge(quantity_segment1_by_pack, quantity_other_by_pack,␣
        ↪left_index=True, right_index=True, suffixes=('_targetSegment', '_other'))
      pack_proportions['affinityToPack'] = pack_proportions['PROD_QTY_targetSegment']␣
        ↪/ pack_proportions['PROD_QTY_other']

      # Display pack proportions sorted by affinity
      pack_proportions.sort_values(by='affinityToPack', ascending=False)
```

```
[32]:            PROD_QTY_targetSegment  PROD_QTY_other  affinityToPack
      PACK_SIZE
      270                      0.031829        0.025073        1.269456
      330                      0.061284        0.050116        1.222842
      380                      0.032160        0.026481        1.214455
      134                      0.119420        0.100542        1.187764
      110                      0.106280        0.089709        1.184728
      210                      0.029124        0.025098        1.160386
      135                      0.014769        0.013063        1.130551
      250                      0.014355        0.012769        1.124201
      170                      0.080773        0.080911        0.998289
      150                      0.157598        0.163270        0.965261
      175                      0.254990        0.269758        0.945252
      165                      0.055652        0.062210        0.894581
      190                      0.007481        0.012431        0.601825
      180                      0.003589        0.006061        0.592083
      160                      0.006404        0.012362        0.518093
      90                       0.006349        0.012569        0.505163
      125                      0.003009        0.006031        0.498902
      200                      0.008972        0.018639        0.481342
      70                       0.003037        0.006317        0.480735
      220                      0.002926        0.006590        0.444005
```