quantium

Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

Author: Albert Bagdasarov

Exploratory data analysis & data Preparation

Loading data.

```
purchases_behaviour = pd.read_csv("datasets/QVI_purchase_behaviour.csv")
transactions_data = pd.read_csv("datasets/QVI_transaction_data.csv")
```

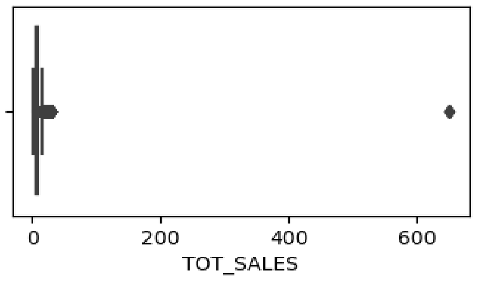
Let's start with checking our datasets for missing or invalid values.

```
purchases_behaviour.isnull().sum()
transactions_data.isnull().sum()
```

Let's check datasets for outliers - Using Standard Deviation (Z-score Method)

```
mean = transactions_data['TOT_SALES'].mean()
std = transactions_data['TOT_SALES'].std()
# Define outlier threshold
z_scores = (transactions_data['TOT_SALES'] - mean) / std
outliers = transactions_data[np.abs(z_scores) > 4]
outliers
```

```
plt.figure(figsize=(4, 2)) sns.boxplot(x=transactions_data['TOT_SALES'])
plt.show()
```

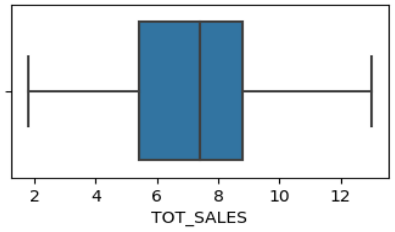


Based on the Z-score method and the plot, there are several outliers that significantly stand out from the rest of the instances

```
# ISO-Forest Machine Learning appoach to find outliers
iso_forest = IsolationForest(contamination=0.009)  # Adjust contamination level
transactions_data['anomaly'] = iso_forest.fit_predict(transactions_data[['TOT_SALES']]
outliers = transactions_data[transactions_data['anomaly'] == -1]
outliers
```

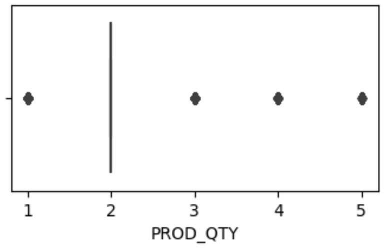
```
transactions_data = transactions_data[transactions_data['anomaly'] != -1]
transactions_data.drop(columns=['anomaly'], inplace=True)  # Remove the anomaly column
```

```
plt.figure(figsize=(4, 2)) sns.boxplot(x=transactions_data['TOT_SALES'])
plt.show()
```



Seems like we are done with outliers in TOT_SALES column.Let's check the PROD_QTY which may also have some outliers

```
plt.figure(figsize=(4, 2)) sns.boxplot(x=transactions_data['PROD_QTY'])
plt.show()
```



No significant outliers have been detected. Let's move one.

- Let's check the data types of each column we have in both datasets

```
print(transactions_data.dtypes)
```

```
print(purchases_behaviour.dtypes)
```

Column 'DATE' has a wrong type, which is int64.

```
transactions_data['DATE'] = pd.to_datetime(transactions_data['DATE'],
origin='1899-12-30', unit = 'D')
```

For the next step Let's look at the Product name column.

Since we need ony Chips. Let's remove the Salsa products

```
transactions_data = transactions_data[~transactions_data['PROD_NAME'].str.contains('salsa,
case=False, na=False)
```

```
agg_products = transactions_data.groupby('PROD_NAME').agg({
     'PROD_QTY': ['min', 'max', 'mean'], # Min, Max, Mean for quantity
     'TOT_SALES': ['min', 'max', 'mean'] # Min, Max, Mean for sales
})
agg_products
```

	PROD_QTY			TOT_SALES		
	min	max	mean	min	max	mean
PROD_NAME						
Burger Rings 220g	1	5	1.898977	2.3	11.5	4.367647
CCs Nacho Cheese 175g	1	5	1.895194	2.1	10.5	3.979907
CCs Original 175g	1	5	1.902246	2.1	10.5	3.994716
CCs Tasty Cheese 175g	1	5	1.877843	2.1	10.5	3.943470
Cheetos Chs & Bacon Balls 190g	1	5	1.893847	3.3	16.5	6.249696
...	...	...	...	...	...	...
WW Original Corn Chips 200g	1	5	1.889632	1.9	9.5	3.590301
WW Original Stacked Chips 160g	1	5	1.884331	1.9	9.5	3.580229
WW Sour Cream &OnionStacked Chips 160g	1	5	1.889413	1.9	9.5	3.589885
WW Supreme Cheese Corn Chips 200g	1	5	1.880053	1.9	9.5	3.572101
Woolworths Cheese Rings 190g	1	5	1.894459	1.8	9.0	3.410026

105 rows × 6 columns

Top Five most popular Chips Brands

```
transactions_data.groupby('PROD_NAME')['PROD_NAME'].count()[:5]
```

```
PROD_NAME
Burger Rings 220g                   1564
CCs Nacho Cheese    175g            1498
CCs Original 175g                   1514
CCs Tasty Cheese    175g            1539
Cheetos Chs & Bacon Balls 190g      1479
Name: PROD_NAME, dtype: int64
```

Let's Filter out the the dataset based on the number of transactions per loyalty card number

```
transcat_counts = transactions_data.groupby('LYLTY_CARD_NBR')['LYLTY_CARD_NBR'].count()
transcat_counts.unique()
```

Let's count number of transactions by date.

```
transcat_counts_by_date = transactions_data.groupby('DATE')['LYLTY_CARD_NBR'].count()
transcat_counts_by_date
```

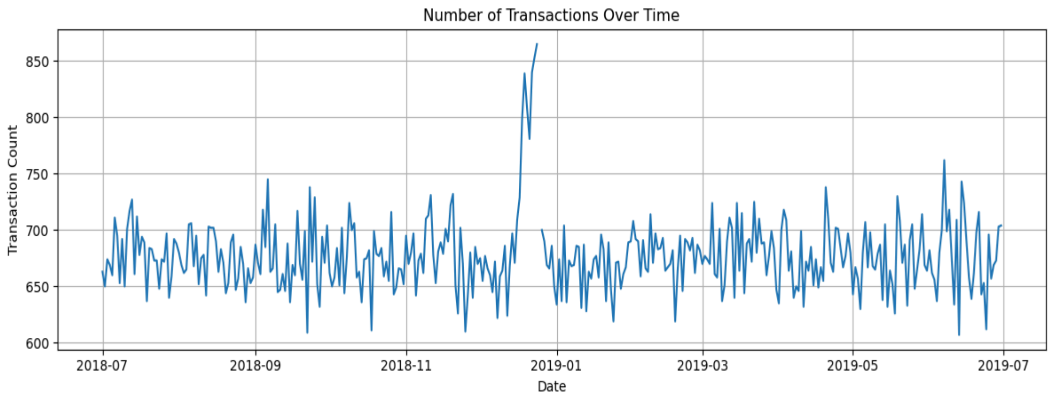
There's only 364 rows, meaning only 364 dates which indicates a missing date.

Let's create a sequence of dates from 1 Jul 2018 to 30 Jun 2019 and use this to create a chart of number of transactions over time to find the missing date.

```
# Step 1: Create a sequence of dates from 1st July 2018 to 30th June 2019
date_seq = pd.date_range(start="2018-07-01", end="2019-06-30", freq="D")
date_seq = pd.DataFrame(date_seq, columns=['DATE'])
```

```
# Step 2: Merge the created DataFrame with our transactions by date dataFrame
merged_data = pd.merge(date_seq,transcat_counts_by_date, on='DATE', how='left')
```

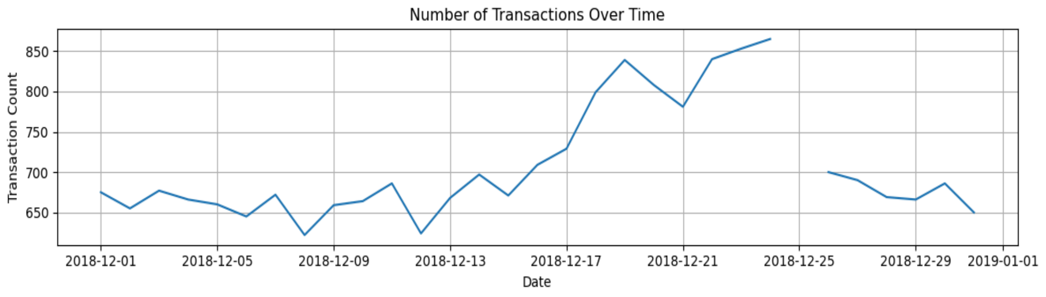
```
# Step 3: Plot the number of transactions over time
plt.figure(figsize=(12, 4))
plt.plot(merged_data['DATE'], merged_data['transaction_count'])
plt.title("Number of Transactions Over Time") plt.xlabel("Date")
plt.ylabel("Transaction Count")
plt.grid(True)
plt.tight_layout()
plt.show()
```



We can see that there is an increase in purchases in December and a break in late December. Let's zoom in on this.

```
# Extracting the December from the merged dataFrame
december_data = merged_data[(merged_data['DATE'] >= '2018-12-01')
& (merged_data['DATE'] <= '2018-12-31')]
```

```
plt.figure(figsize=(12, 3))
plt.plot(december_data['DATE'], december_data['transaction_count'])
plt.title("Number of Transactions Over Time")
plt.xlabel("Date")
plt.ylabel("Transaction Count")
plt.grid(True)
plt.tight_layout()
plt.show()
```



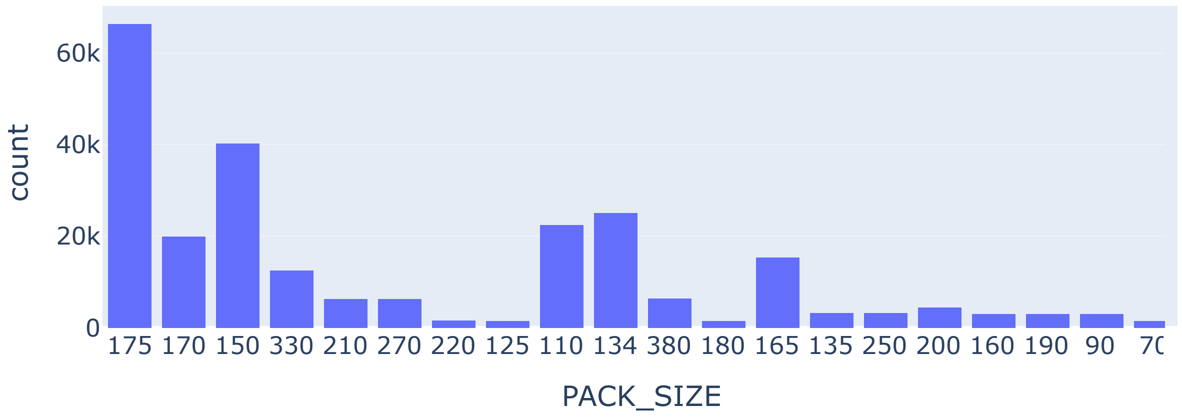
We can see that the increase in sales occurs in the lead-up to Christmas and that there are zero sales on Christmas day itself. This is due to shops being closed on Christmas day.

Let's create a new feature called 'pack size' which will take the weight portion of the prodcut extracted from PROD_NAME

```
transactions_data['PACK_SIZE'] = transactions_data['PROD_NAME'].apply(
lambda x: x[-4:-1])
transactions_data['PACK_SIZE'] = transactions_data['PROD_NAME'].str.extract('(\d+)')
transactions_data['PACK_SIZE'].unique()
```

```
array(['175', '170', '150', '330', '210', '270', '220', '125', '110',
       '134', '380', '180', '165', '135', '250', '200', '160', '190',
       '90', '70'], dtype=object)
```

```
fig = px.histogram(transactions_data, x='PACK_SIZE')
fig.update_layout( width = 700, height = 300) fig.show(renderer='iframe')
```



- the most frequent package size is 175 grams
- the highest one is 380 grams
- the smallest is 70 grams

Let's create a new feature 'BRAND' which will take the Chip's brand name from PROD_NAME column

```
transactions_data['BRAND'] = transactions_data['PROD_NAME'].str.split().str[0]
```

Some of the brand names look like they are of the same brands - such as RED and RRD, which are both Red Rock Deli chips. Dorito and Doritos. Snbts and Sunbites Let's combine these together.

```
transactions_data['BRAND'] = transactions_data['BRAND'].replace({'RED': 'RRD',
'Dorito':'Doritos', 'Snbts':'Sunbites'})
transactions_data['BRAND'].unique()
```

```
array(['Natural', 'CCs', 'Smiths', 'Kettle', 'Grain', 'Doritos',
       'Twisties', 'WW', 'Thins', 'Burger', 'NCC', 'Cheezels', 'Infzns',
       'Red', 'Pringles', 'Infuzions', 'Smith', 'GrnWves', 'Tyrrells',
       'Cobs', 'French', 'RRD', 'Tostitos', 'Cheetos', 'Woolworths',
       'Sunbites'], dtype=object)
```

Now that we are happy with the transaction dataset, let's have a look at the customer dataset.

```
# Let's Merge two datasets
merged_data = pd.merge(transactions_data, purchases_behaviour, on = "LYLTY_CARD_NBR",
how = 'left')
merged_df.isnull().sum()
```

Data analysis on customer segments

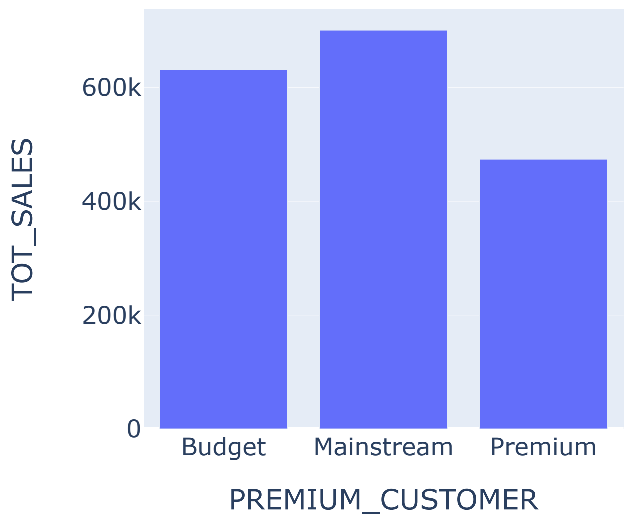
Now that the data is ready for analysis, we can define some metrics of interest to the client:

- Who spends the most on chips (total sales), describing customers by lifestage and how premium their general purchasing behaviour is
- How many customers are in each segment
- How many chips are bought per customer by segment
- What's the average chip price by customer segment
- The customer's total spend over the period and total spend for each transaction to understand what proportion of their grocery spend is on chips
- Proportion of customers in each customer segment overall to compare against the mix of customers who purchase chips

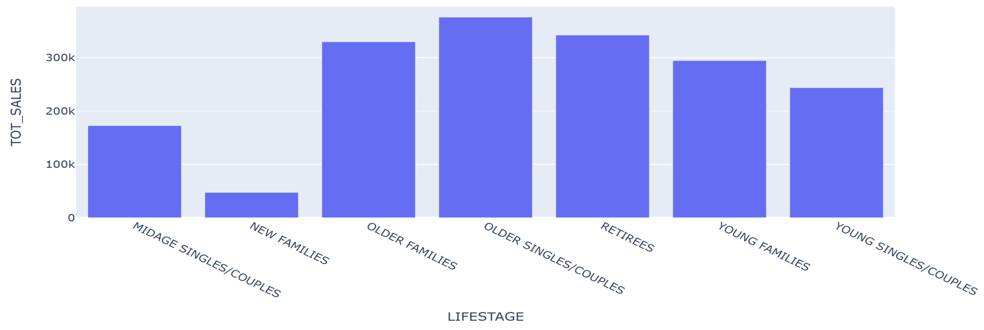
Let's start with calculating total sales by LIFESTAGE and PREMIUM_CUSTOMER and plotting the split by these segments to describe which customer segment contribute most to chip sales.

```
total_sales_by_permium = merged_df.groupby('PREMIUM_CUSTOMER').agg({'TOT_SALES':'sum'}.
reset_index()

total_sales_by_lifestage = merged_df.groupby('LIFESTAGE').agg({'TOT_SALES':'sum'}).
reset_index()

total_sales_by_customers = merged_df.groupby(['LIFESTAGE','PREMIUM_CUSTOMER'])[
'TOP_SALES'].sum().reset_index()
```

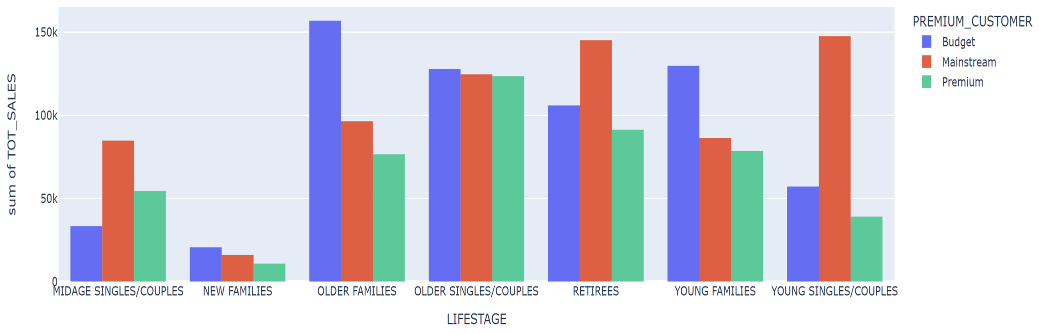
```
fig = px.bar(total_sales_by_permium , x='PREMIUM_CUSTOMER', y='TOT_SALES')
fig.update_layout( width = 400, height = 350)
fig.show(renderer='iframe')
```



```
fig = px.bar(total_sales_by_lifestage, x='LIFESTAGE', y='TOT_SALES')
fig.update_layout( width = 1000, height = 450) fig.show(renderer='iframe')
```



```
fig = px.histogram(total_sales_by_customers , x="LIFESTAGE", y="TOT_SALES",
color='PREMIUM_CUSTOMER', barmode='group', height=400)
fig.show(renderer='iframe')
```

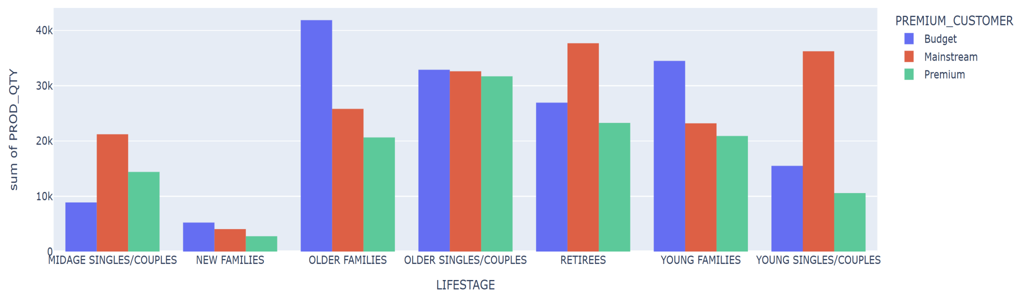


- The highest sales in Premium Segment has 'OLDER SINGLES/COUPLES' category. Moreover this is most balanced category of customers.
- The highest sales in Budget Segment has 'OLDER FAMILIES' category.
- The lowest sales in all three Segments has 'NEW FAMILIES'!

Higher sales may also be driven by more units of chips being bought per customer. Let's have a look at this next

```
avg_units_by_customers = merged_df.groupby(['LIFESTAGE','PREMIUM_CUSTOMER'])
['PROD_QT'].sum().reset_index()
avg_units_by_customers
```

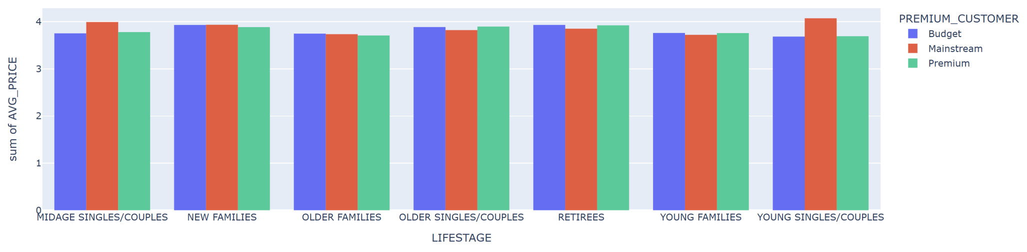
```
fig = px.histogram(avg_units_by_customers, x="LIFESTAGE", y="PROD_QTY",
color='PREMIUM_CUSTOMER', barmode='group', height=400)
fig.show(renderer='iframe')
```



Older families and young families in general buy more chips per customer
Let's also investigate the average price per unit chips bought for each customer segment as this is also a driver of total sales.

```python
# Average price per unit by LIFESTAGE and PREMIUM_CUSTOMER
#Step 1: Calculate total sales and total quantity for each segment
segment_sales = merged_df.groupby(['LIFESTAGE', 'PREMIUM_CUSTOMER']).agg(
{'TOT_SALES':'sum','PROD_QTY':'sum'}).reset_index()
#Step 2: Calculate average price per unit chips
segment_sales['AVG_PRICE'] = segment_sales['TOT_SALES'] / segment_sales['PROD_QTY']

segment_sales
```

```python
fig = px.histogram(segment_sales, x="LIFESTAGE", y="AVG_PRICE",
color='PREMIUM_CUSTOMER', barmode='group', height=400)
fig.show(renderer='iframe')
```



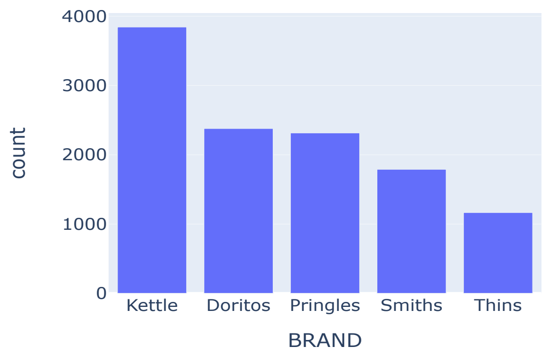
As the difference in average price per unit isn't large, we can check if this difference is statistically different
Perform an independent t-test between mainstream vs premium and budget midage and young singles and couples

```python
# Filter for mainstream and premium customers
mainstream = merged_df[merged_df['PREMIUM_CUSTOMER'] == 'Mainstream']['PROD_QTY']
premium = merged_df[merged_df['PREMIUM_CUSTOMER'] == 'Premium']['PROD_QTY']
# Perform t-test
t_stat, p_value = ttest_ind(mainstream, premium, equal_var=False)
print(f"T-test Mainstream vs Premium: t-stat={t_stat:.3f}, p-value={p_value:.3f}")
```

```
T-test Mainstream vs Premium: t-stat=-2.411, p-value=0.016
The t-test results in a p-value of 0.016, i.e. the unit price for mainstream, young and mid-age singles and couples
not significantly higher than that of budget or premium, young and midage singles and couples.

We might want to target customer segments that contribute the most to sales to retain them or further increase sales.
Let's look at Mainstream - young singles/couples.
For instance, let's find out if they tend to buy a particular brand of chips
```

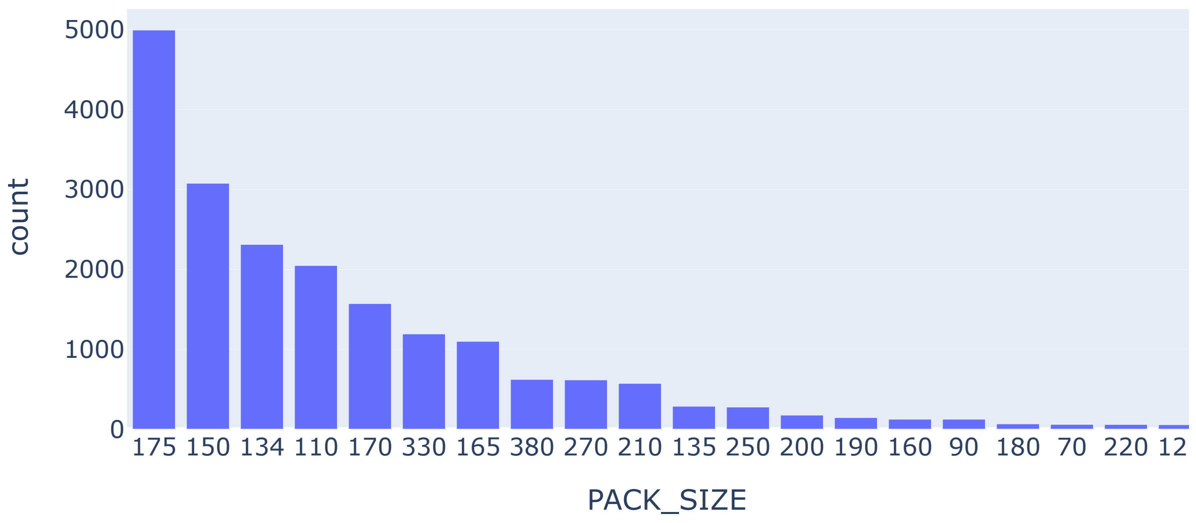
```
young_mainstream = merged_df[(merged_df['LIFESTAGE'] == 'YOUNG SINGLES/COUPLES') &
(merged_df['PREMIUM_CUSTOMER'] == 'MAINSTREAM')]

top_brands = young_mainstream.groupby('BRAND')['BRAND'].count().
sort_values(ascending=False)
```

```
fig = px.bar(top_brands, x='BRAND', y='count')
fig.update_layout( width = 450, height = 350)
fig.show(renderer='iframe')
```



Kettle is the most popular brand among the customer's category who buy most of Chips.
Let's also find out if our target segment tends to buy larger packs of chips

```
pack_size = young_mainstream.groupby('PACK_SIZE')['PACK_SIZE'].count().sort_values(
ascending = False).reset_index(name='count')
fig = px.bar(pack_size , x='PACK_SIZE', y='count')
fig.update_layout( width = 700, height = 350)
fig.show(renderer='iframe')
```


count
PACK_SIZE