```
Start coding or generate with AI.
```

Name : Trakshay Balagotra

Contact Number : +91 8308103278

Mail ID : trakshay.balagotra@outlook.com

Forage Quantium Data Analytics Task 2 : Experimenting and Uplift Testing

1. Importing the necessary dependencies

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
```

2. Reading the dataset

```
data = pd.read_csv('QVI_data.csv')
```

```
data.head()
```

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PREMIUM_CUSTOMER
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATURAL	YOUNG SINGLES/COUPLES	Premium
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	RRD	YOUNG SINGLES/COUPLES	Mainstream
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNWVES	YOUNG FAMILIES	Budget
		2019-03-				Natural ChipCo Hony Sov						

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264834 entries, 0 to 264833
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   LYLTY_CARD_NBR    264834 non-null  int64
 1   DATE              264834 non-null  object
 2   STORE_NBR         264834 non-null  int64
 3   TXN_ID            264834 non-null  int64
 4   PROD_NBR          264834 non-null  int64
 5   PROD_NAME         264834 non-null  object
 6   PROD_QTY          264834 non-null  int64
 7   TOT_SALES         264834 non-null  float64
 8   PACK_SIZE         264834 non-null  int64
 9   BRAND             264834 non-null  object
 10  LIFESTAGE         264834 non-null  object
 11  PREMIUM_CUSTOMER  264834 non-null  object
dtypes: float64(1), int64(6), object(5)
memory usage: 24.2+ MB
```

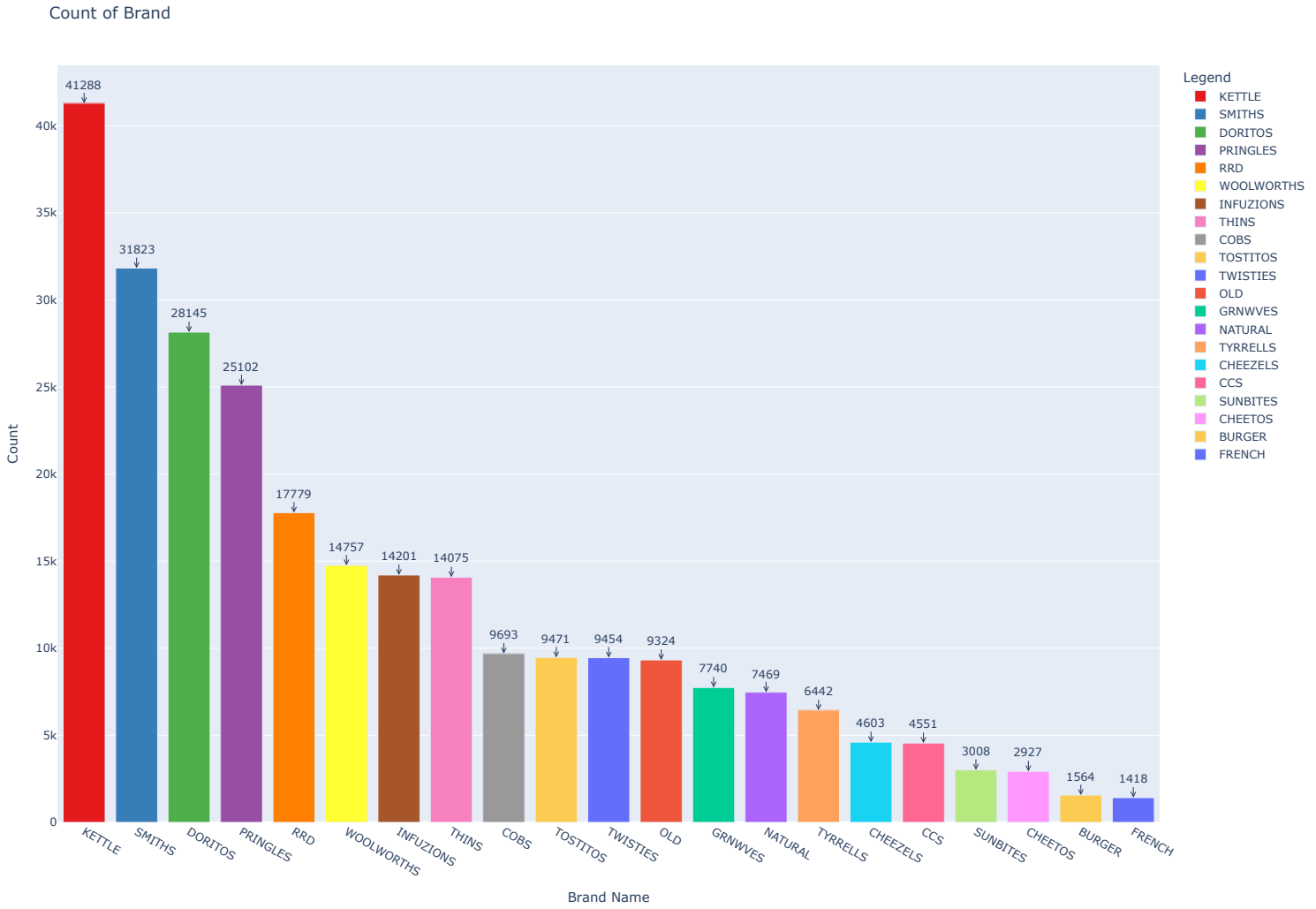
```
data.BRAND.value_counts()
```

BRAND	count
KETTLE	41288
SMITHS	31823
DORITOS	28145
PRINGLES	25102
RRD	17779
WOOLWORTHS	14757
INFUZIONS	14201
THINS	14075
COBS	9693
TOSTITOS	9471
TWISTIES	9454
OLD	9324
GRNWVES	7740
NATURAL	7469
TYRRELLS	6442
CHEEZELS	4603
CCS	4551
SUNBITES	3008
CHEETOS	2927
BURGER	1564
FRENCH	1418

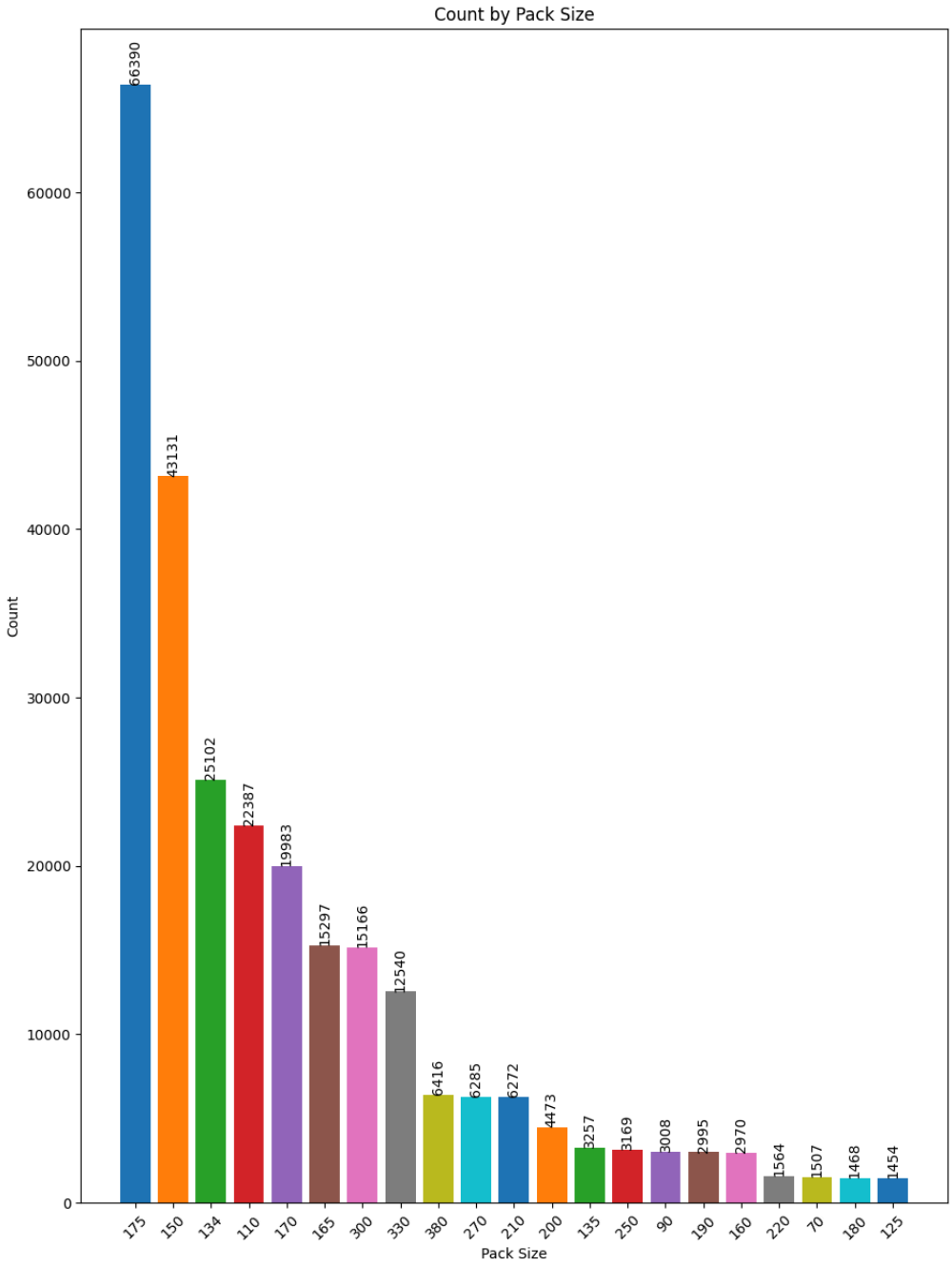
dtype: int64

```
counts = data.BRAND.value_counts()


colors = px.colors.qualitative.Set1[:len(counts)]
```

```
# Create a bar chart using Plotly Express with different colors
fig = px.bar(
    x=counts.index,
    y=counts.values,
    labels={'y': 'Count', 'x':'Brand Name'},
    color=counts.index,
    color_discrete_map={ctype: color for ctype, color in zip(counts.index, colors)},  # Assign custom colors
    title='Count of Brand',
)
for i, count in enumerate(counts.values):
    fig.add_annotation(
        x=counts.index[i],
        y=count,
        text=str(count),
        showarrow=True,
        arrowhead=5,
        ax=0,
        ay=-20,
    )

# Add a legend
fig.update_layout(legend=dict(title=dict(text='Legend')), height = 1000)

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




```
counts = data['PACK_SIZE'].value_counts()

# Plotting the counts
plt.figure(figsize=(11, 15))

for i, (value, count) in enumerate(counts.items()):
    plt.bar(i, count)
    plt.text(i, count + 0.1, str(count), ha='center', va='bottom', rotation=90)

plt.title('Count by Pack Size')
plt.xlabel('Pack Size')
plt.ylabel('Count')
plt.xticks(range(len(counts)), counts.index, rotation=45)

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




```
data['DATE'] = pd.to_datetime(data['DATE'], format='%Y-%m-%d')
```

```
#Extracting Year from DATE column
data['Year'] = data['DATE'].dt.year
```

```
#Extracting Month_name and Year from DATE column
data['Month_Year'] = data['DATE'].dt.strftime('%B %Y')
```

```
#Extracting Month_name from DATE column
data['Month_Name'] = data['DATE'].dt.strftime('%B')
```

```
##Extracting Quater from DATE column
data['Quarter'] = data['DATE'].dt.quarter
```

```
#Extracting Quater and Year from DATE column
data['Quarter_Year'] = data['DATE'].dt.to_period('Q')
```

```
#Extracting WEEKDAY from DATE column
data['Weekday'] = data['DATE'].dt.day_name()
```

```
data.head()
```

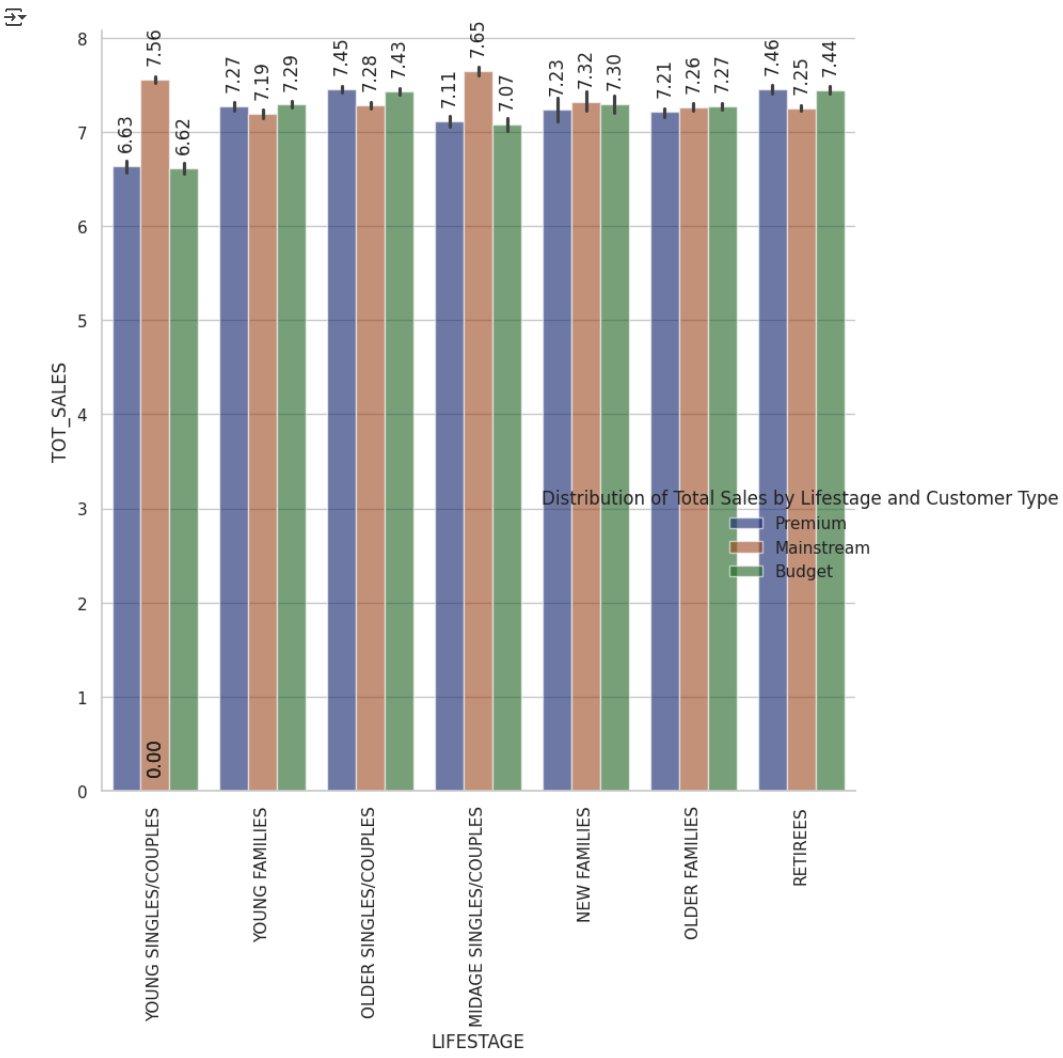
	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PREMIUM_CUSTOMER	Year	Month_Year	Month_Name	Qua
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATURAL	YOUNG SINGLES/COUPLES	Premium	2018	October 2018	October	
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	RRD	YOUNG SINGLES/COUPLES	Mainstream	2018	September 2018	September	
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNWVES	YOUNG FAMILIES	Budget	2019	March 2019	March	
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	1	3.0	175	NATURAL	YOUNG FAMILIES	Budget	2019	March 2019	March	
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	1	1.9	160	WOOLWORTHS	OLDER SINGLES/COUPLES	Mainstream	2018	November 2018	November	

```
#Saving the updated dataset
data.to_csv('updated_data.csv')
data.to_excel('updated_data.xlsx')
```

```
data.head()
```

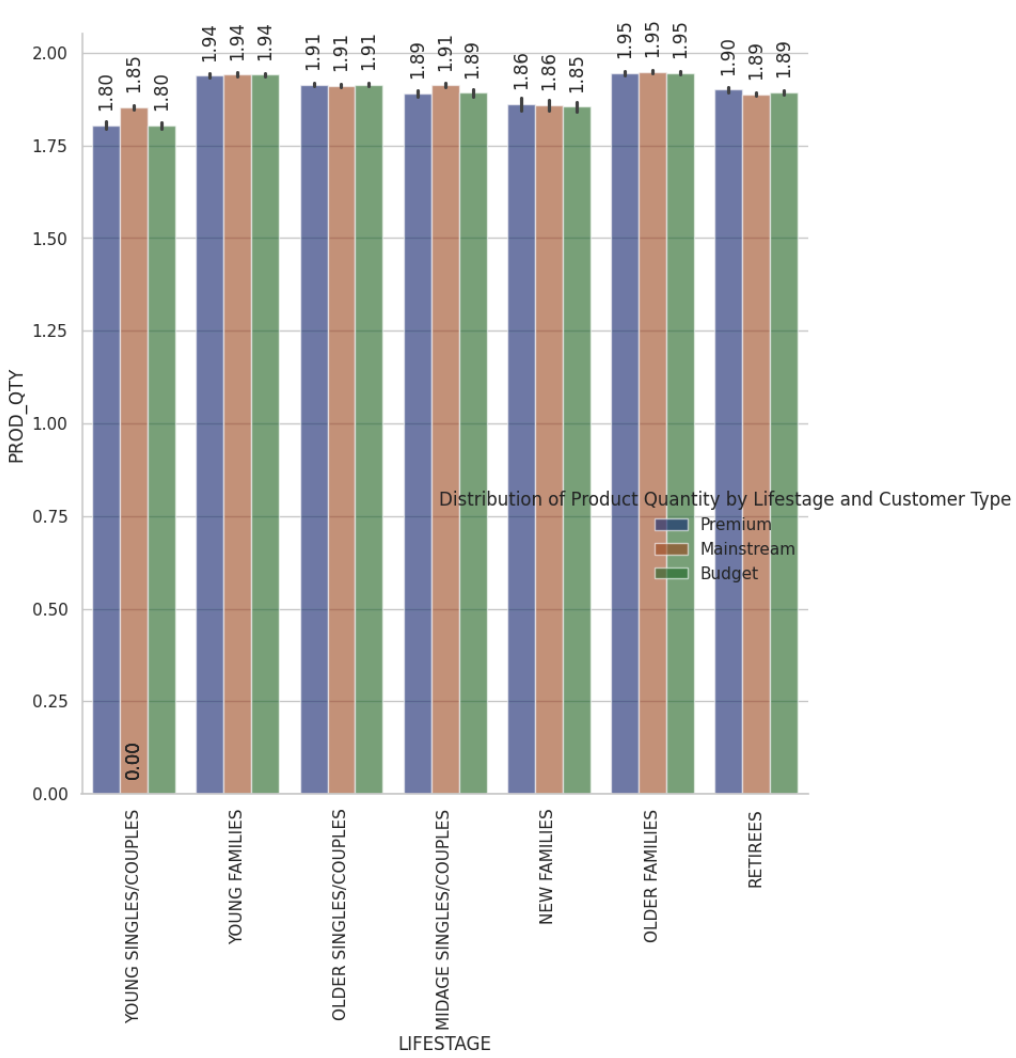
	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	PACK_SIZE	BRAND	LIFESTAGE	PREMIUM_CUSTOMER	Year	Month_Year	Month_Name	Qua
0	1000	2018-10-17	1	1	5	Natural Chip Compny SeaSalt175g	2	6.0	175	NATURAL	YOUNG SINGLES/COUPLES	Premium	2018	October 2018	October	
1	1002	2018-09-16	1	2	58	Red Rock Deli Chikn&Garlic Aioli 150g	1	2.7	150	RRD	YOUNG SINGLES/COUPLES	Mainstream	2018	September 2018	September	
2	1003	2019-03-07	1	3	52	Grain Waves Sour Cream&Chives 210G	1	3.6	210	GRNWVES	YOUNG FAMILIES	Budget	2019	March 2019	March	
3	1003	2019-03-08	1	4	106	Natural ChipCo Hony Soy Chckn175g	1	3.0	175	NATURAL	YOUNG FAMILIES	Budget	2019	March 2019	March	
4	1004	2018-11-02	1	5	96	WW Original Stacked Chips 160g	1	1.9	160	WOOLWORTHS	OLDER SINGLES/COUPLES	Mainstream	2018	November 2018	November	

```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="LIFESTAGE", y="TOT_SALES", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("LIFESTAGE", "TOT_SALES")
g.legend.set_title("Distribution of Total Sales by Lifestage and Customer Type")
```

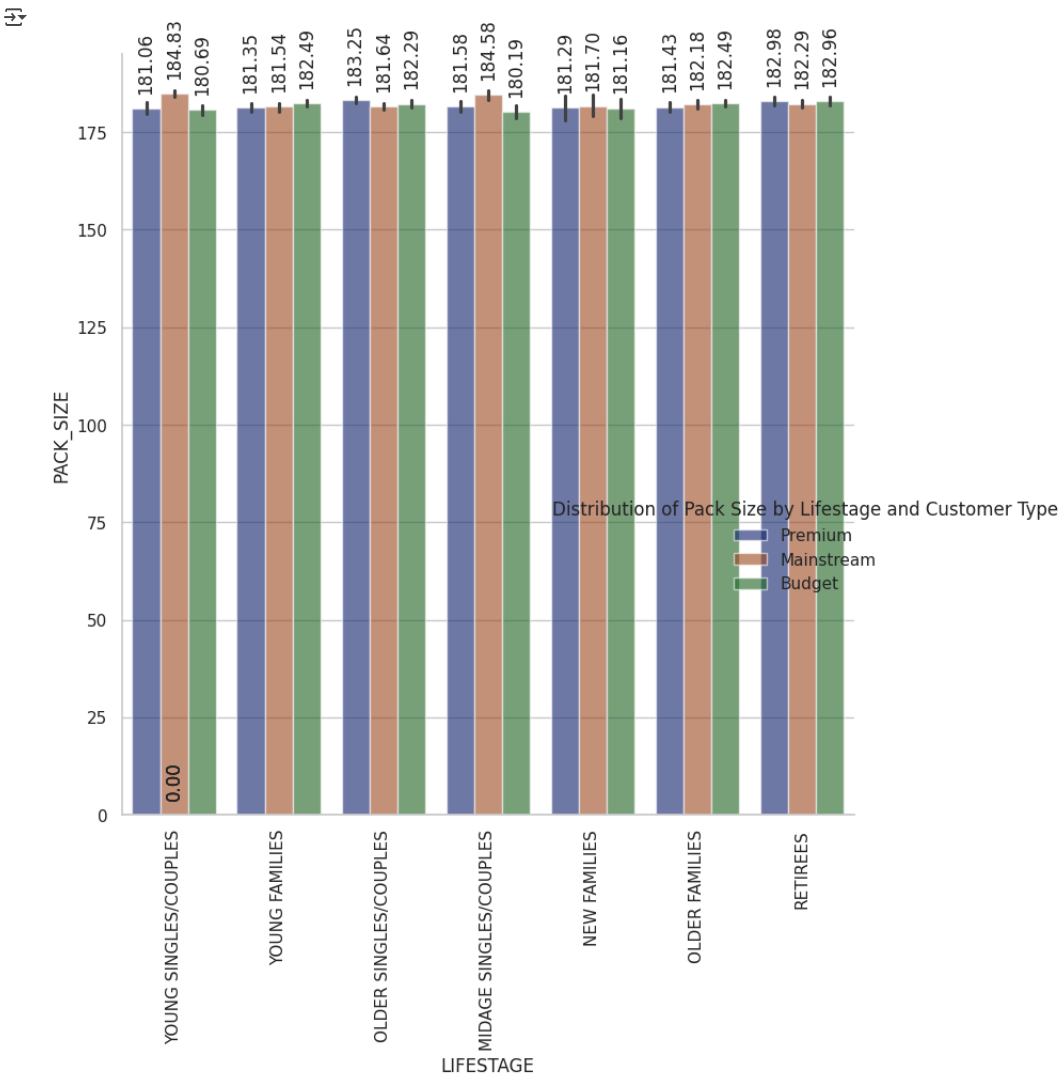



```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="LIFESTAGE", y="PROD_QTY", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("LIFESTAGE", "PROD_QTY")
g.legend.set_title("Distribution of Product Quantity by Lifestage and Customer Type")
```

[+]

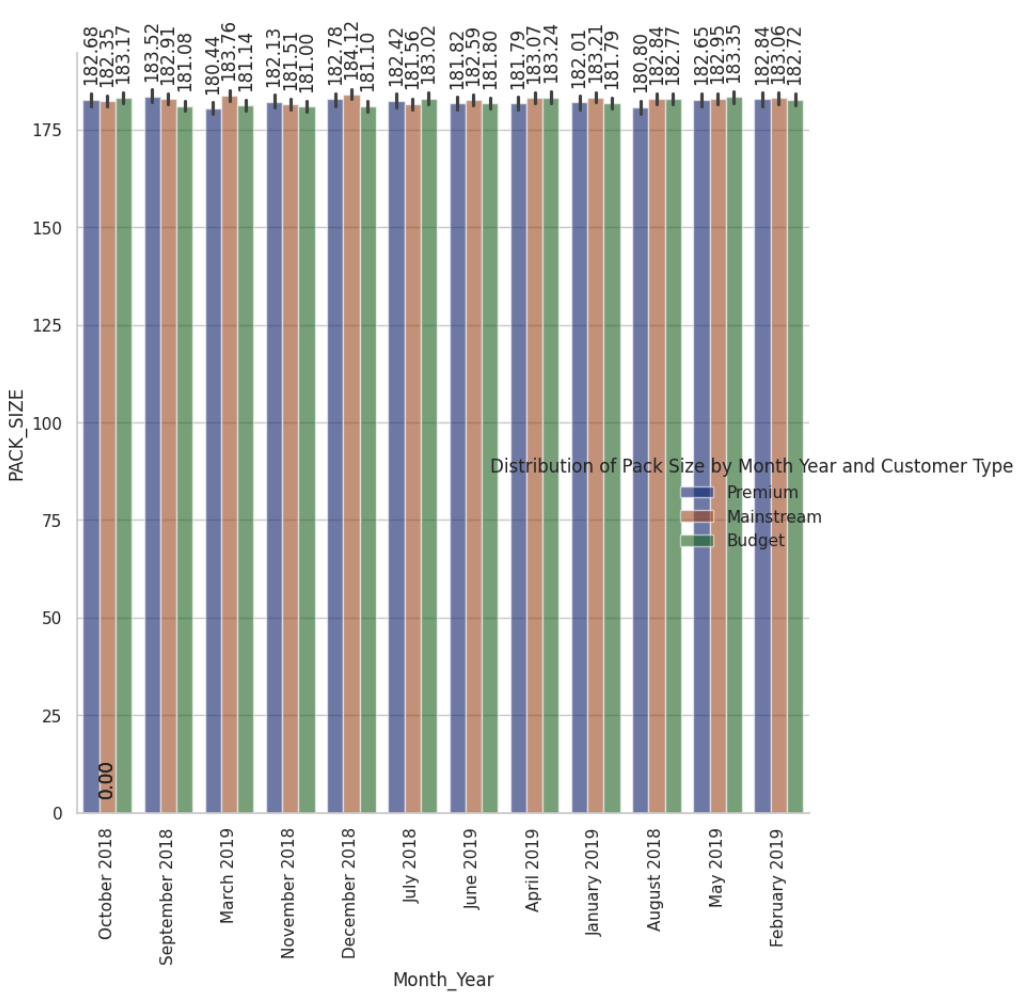



```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="LIFESTAGE", y="PACK_SIZE", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("LIFESTAGE", "PACK_SIZE")
g.legend.set_title("Distribution of Pack Size by Lifestage and Customer Type")
```



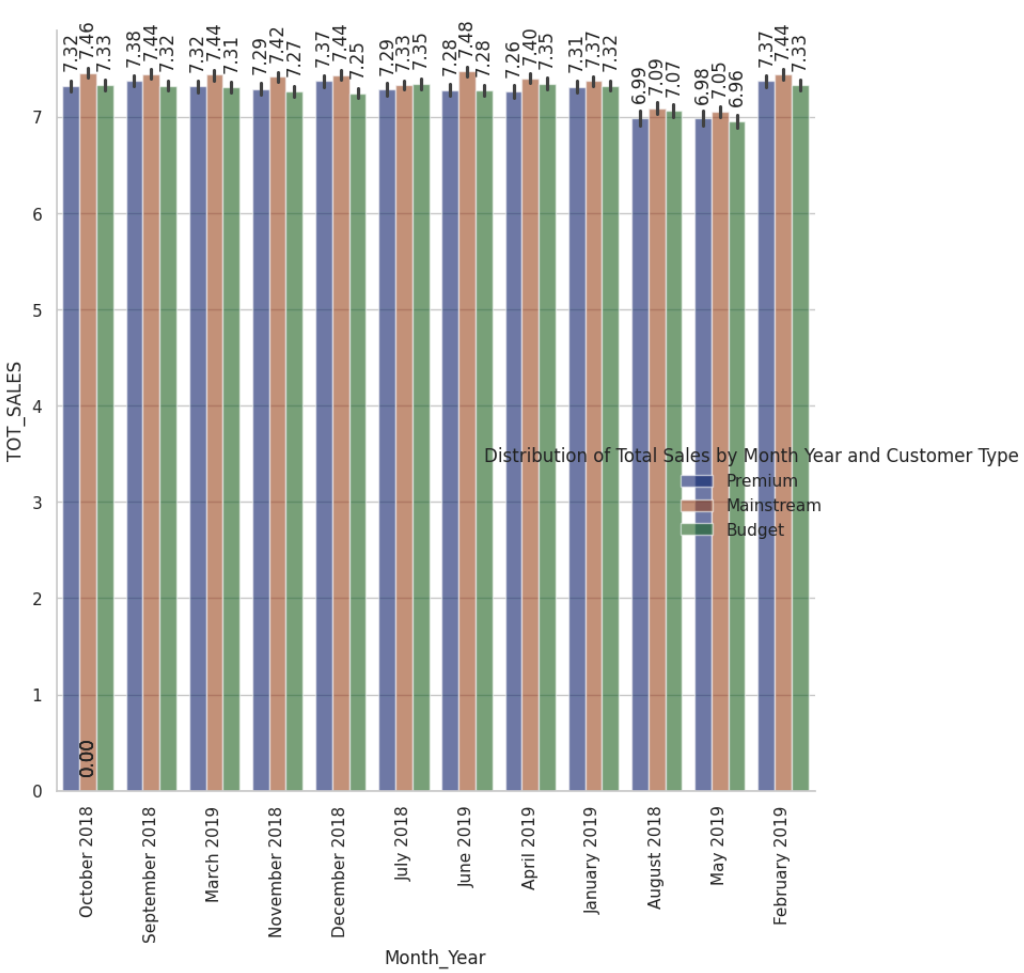

```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Month_Year", y="PACK_SIZE", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Month_Year", "PACK_SIZE")
g.legend.set_title("Distribution of Pack Size by Month Year and Customer Type")
```

[+]

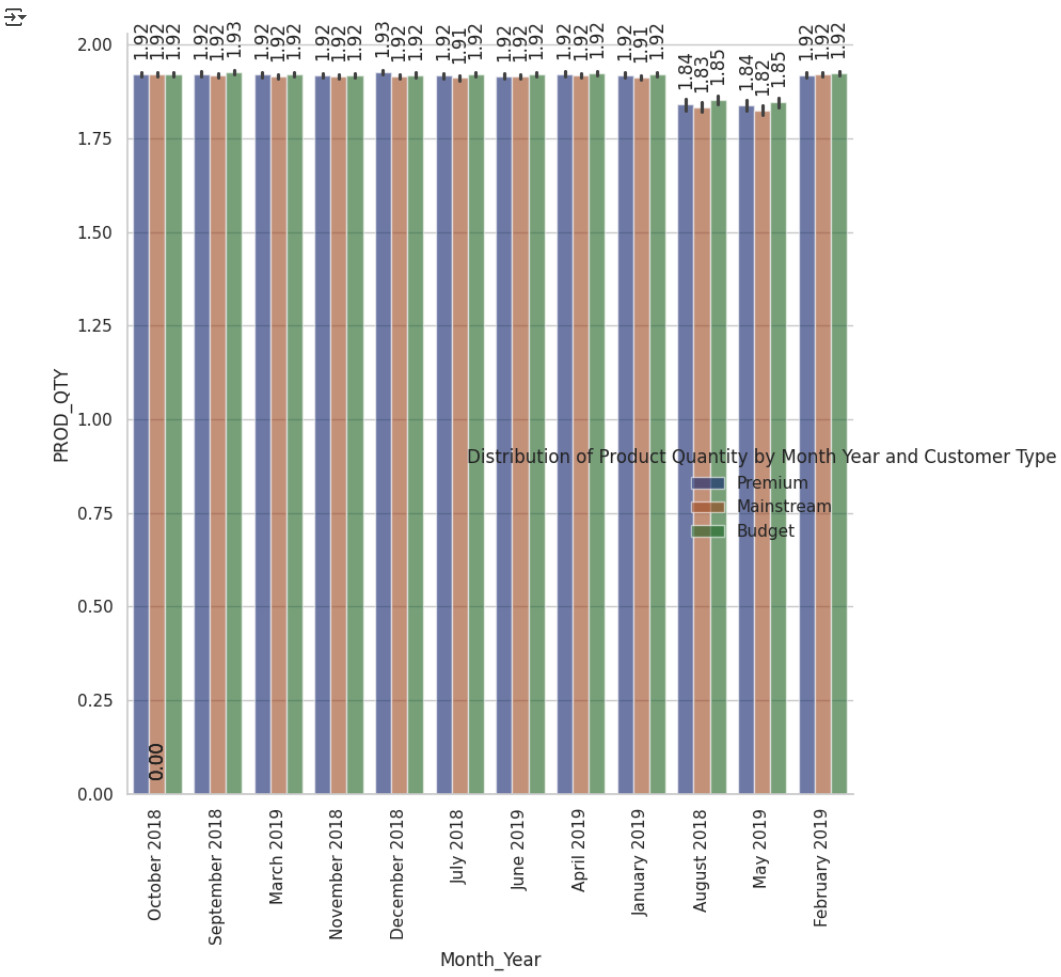



```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Month_Year", y="TOT_SALES", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Month_Year", "TOT_SALES")
g.legend.set_title("Distribution of Total Sales by Month Year and Customer Type")
```

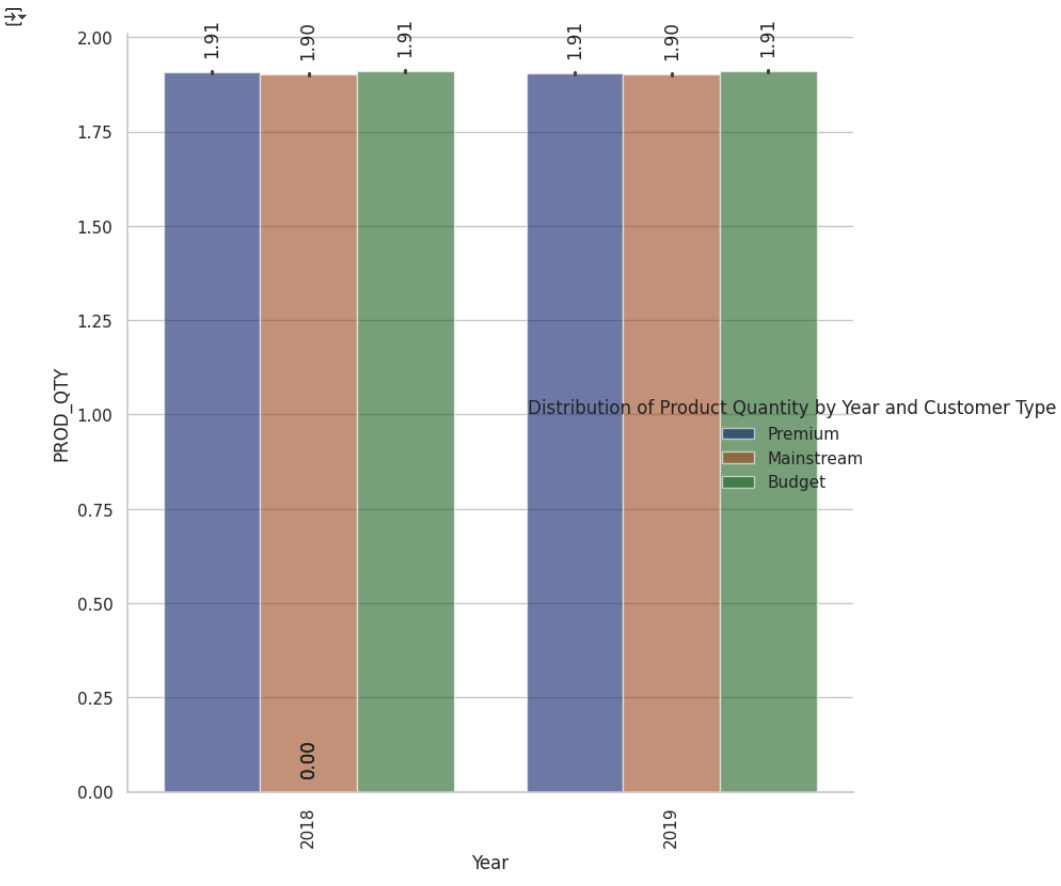
[+]




```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Month_Year", y="PROD_QTY", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Month_Year", "PROD_QTY")
g.legend.set_title("Distribution of Product Quantity by Month Year and Customer Type")
```

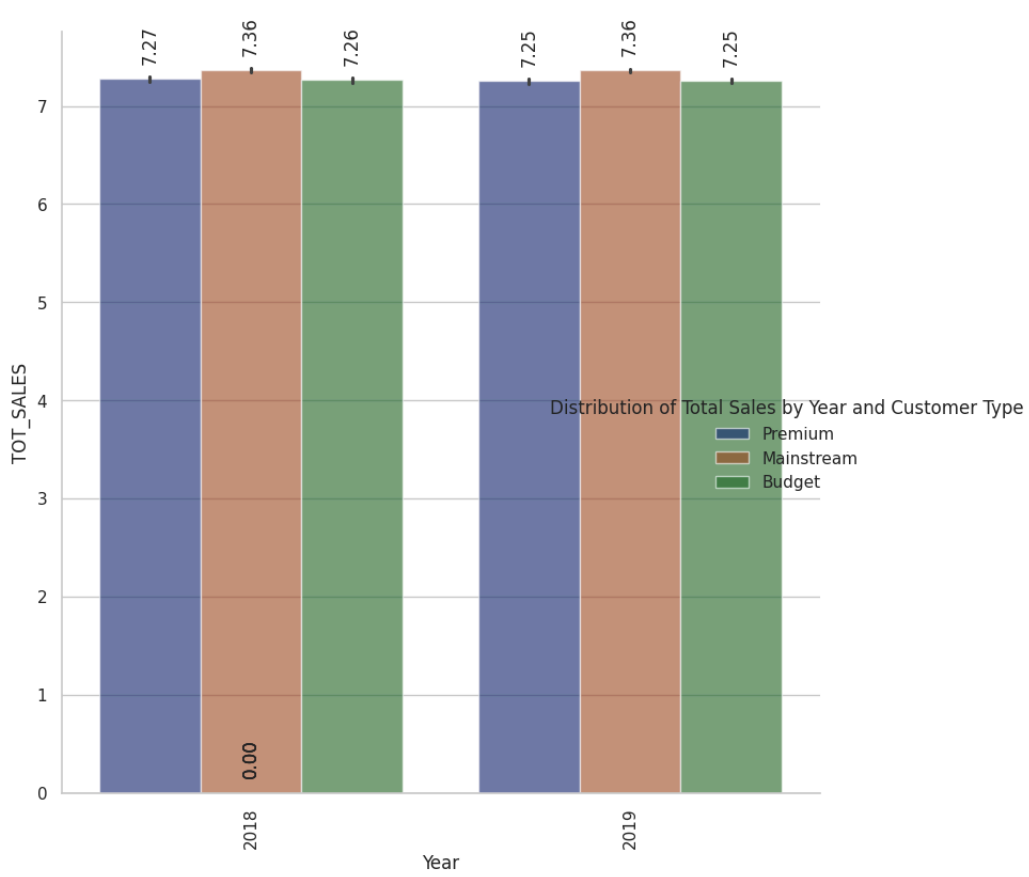



```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Year", y="PROD_QTY", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Year", "PROD_QTY")
g.legend.set_title("Distribution of Product Quantity by Year and Customer Type")
```

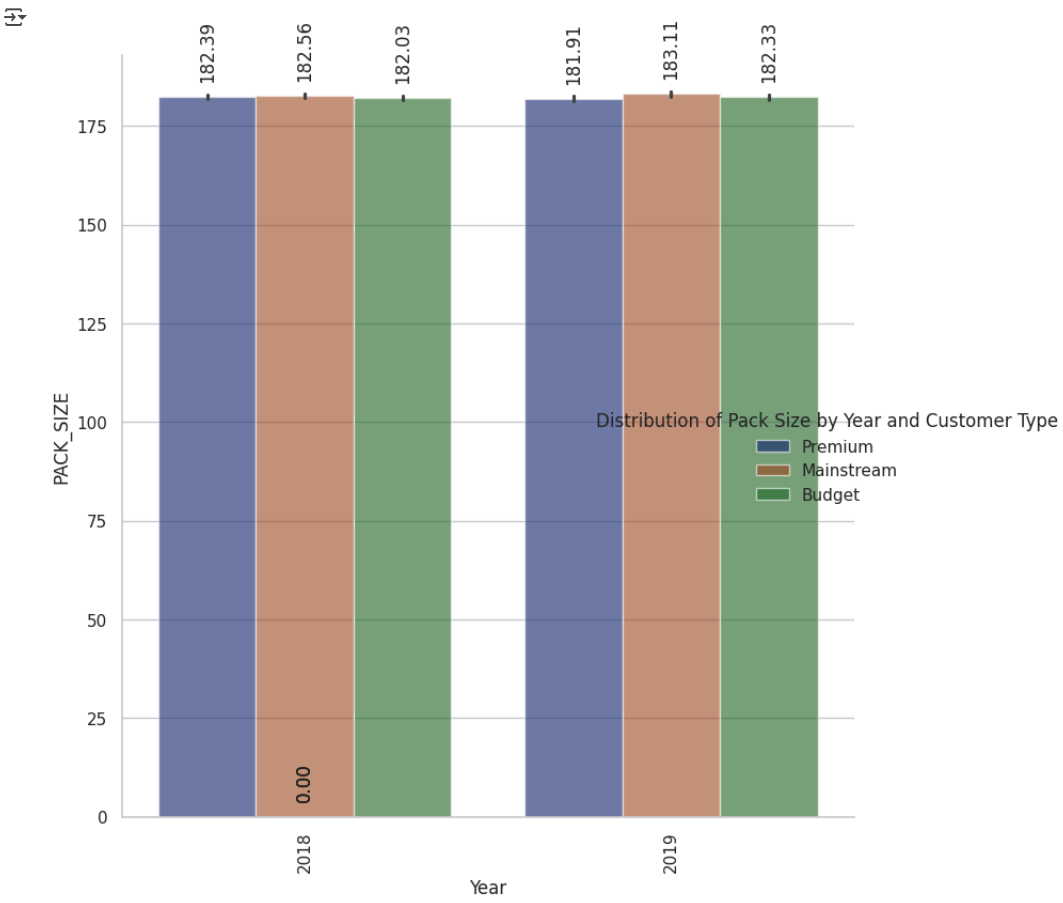


```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Year", y="TOT_SALES", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Year", "TOT_SALES")
g.legend.set_title("Distribution of Total Sales by Year and Customer Type")
```

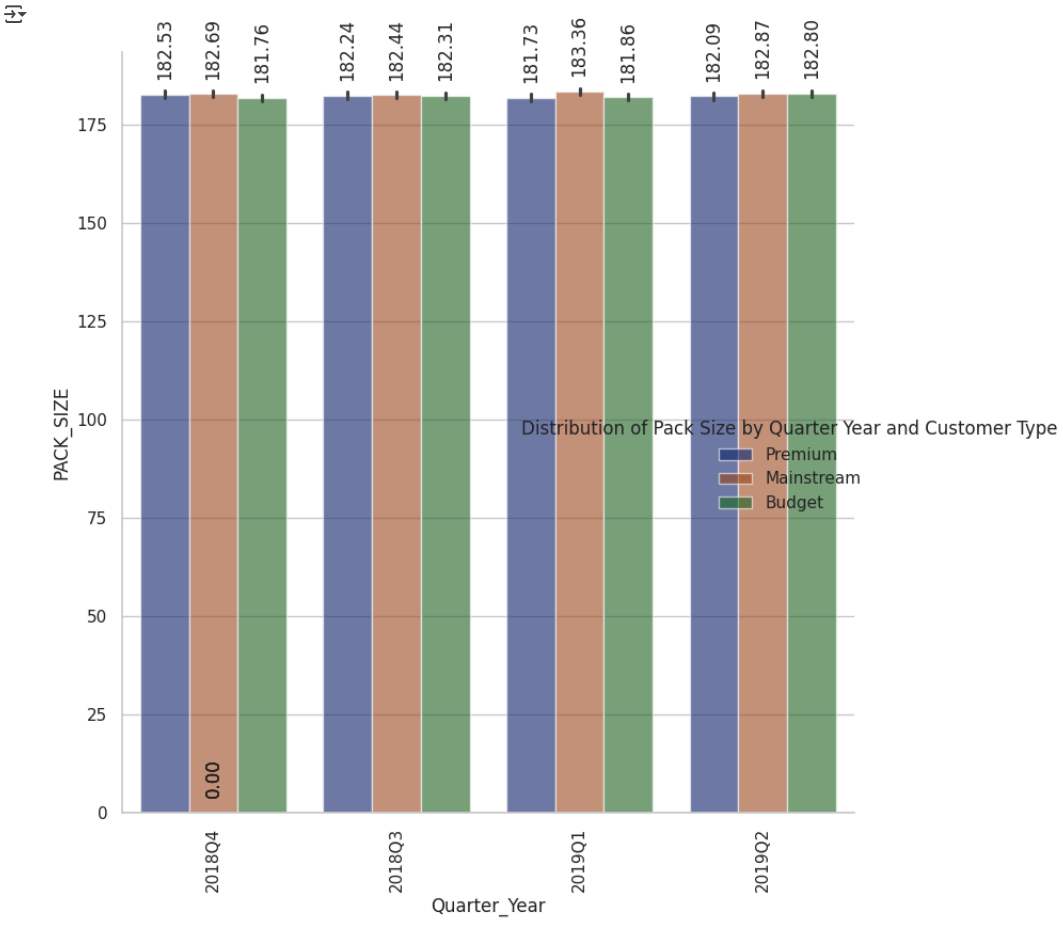
[4]




```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Year", y="PACK_SIZE", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Year", "PACK_SIZE")
g.legend.set_title("Distribution of Pack Size by Year and Customer Type")
```



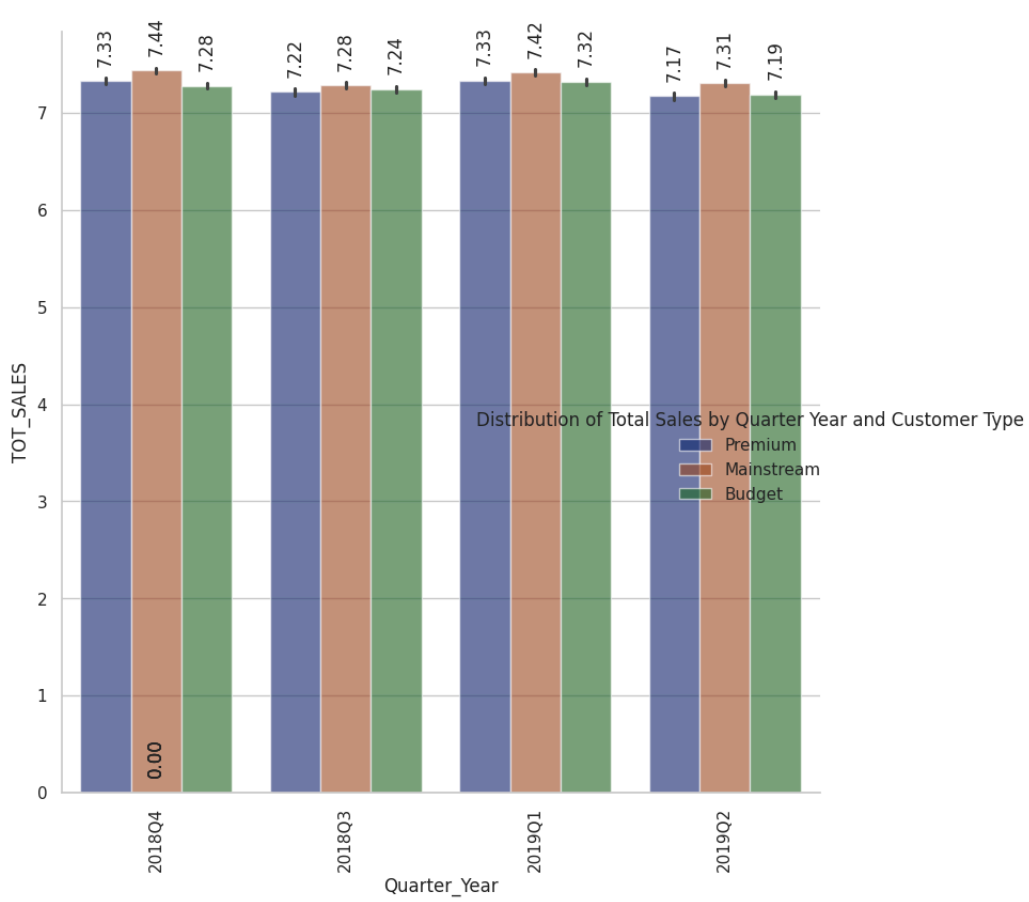

```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Quarter_Year", y="PACK_SIZE", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Quarter_Year", "PACK_SIZE")
g.legend.set_title("Distribution of Pack Size by Quarter Year and Customer Type")
```



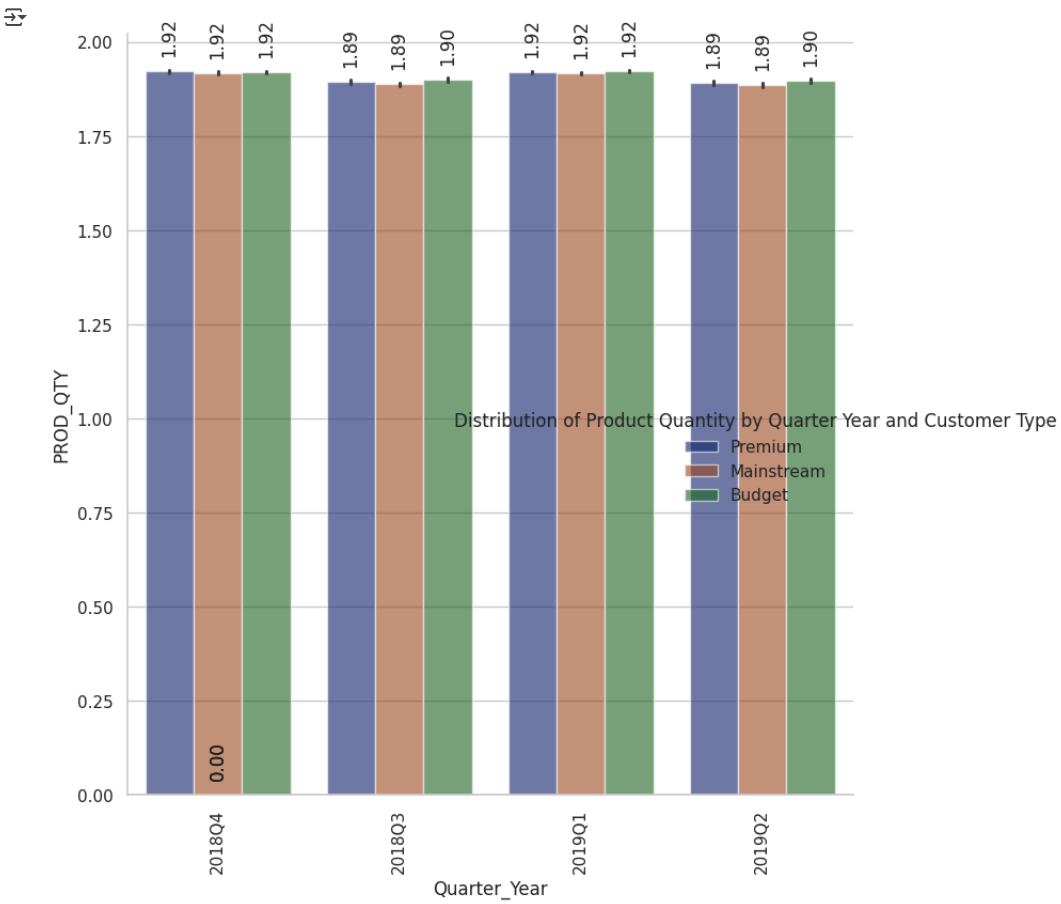

```
sns.set_theme(style="whitegrid")
```

```
g = sns.catplot(
    data=data, kind="bar",
    x="Quarter_Year", y="TOT_SALES", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Quarter_Year", "TOT_SALES")
g.legend.set_title("Distribution of Total Sales by Quarter Year and Customer Type")
```

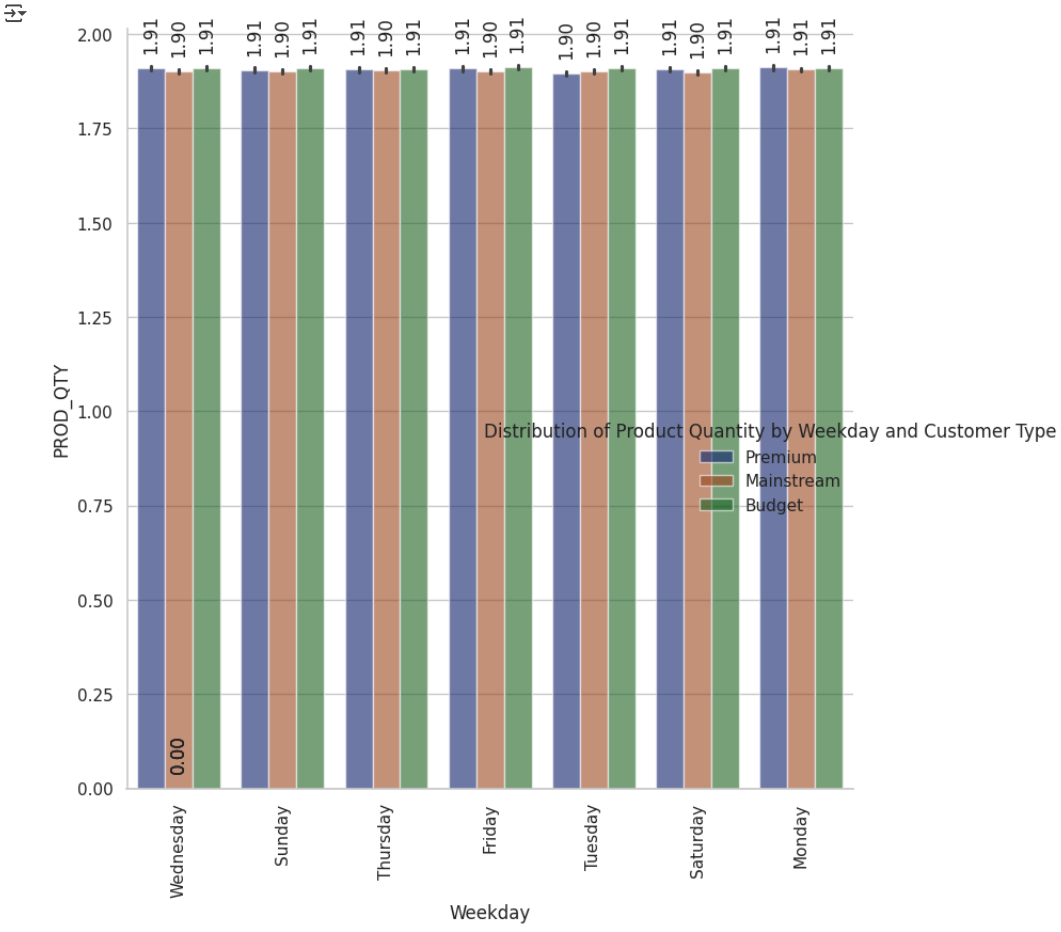
[4]




```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Quarter_Year", y="PROD_QTY", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Quarter_Year", "PROD_QTY")
g.legend.set_title("Distribution of Product Quantity by Quarter Year and Customer Type")
```



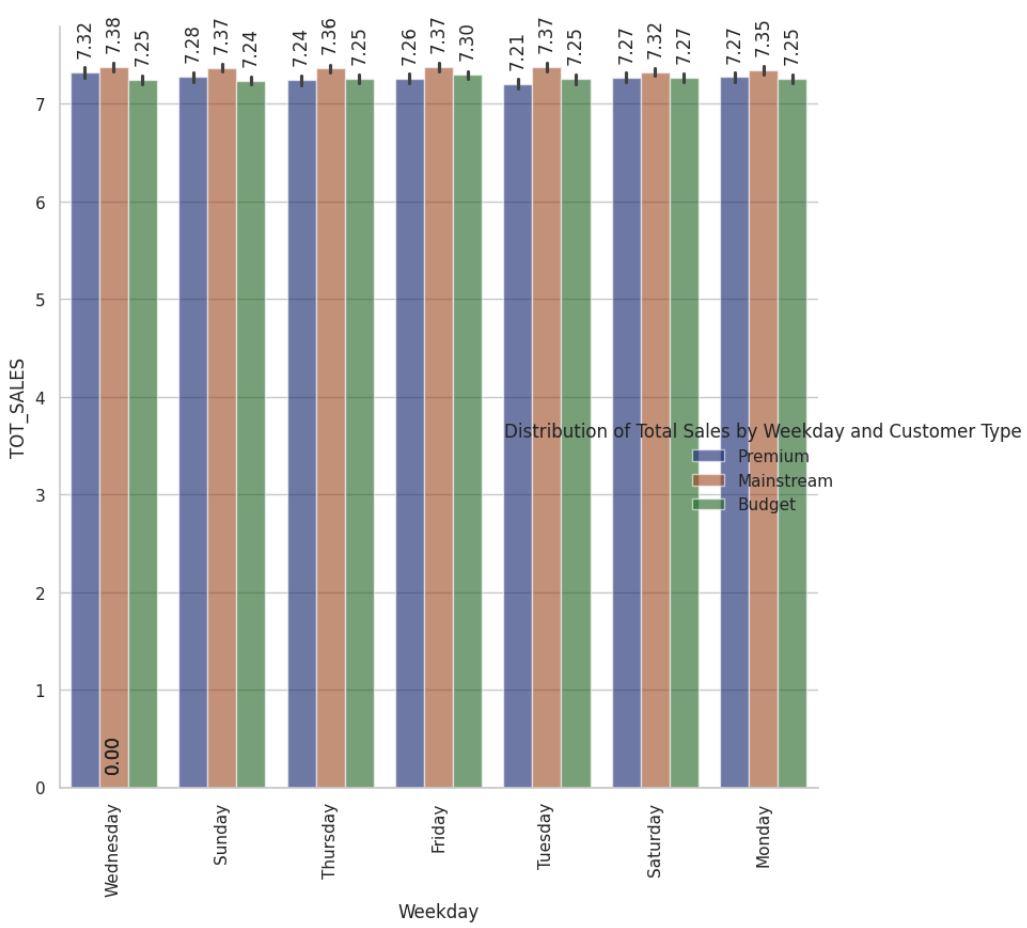

```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Weekday", y="PROD_QTY", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Weekday", "PROD_QTY")
g.legend.set_title("Distribution of Product Quantity by Weekday and Customer Type")
```



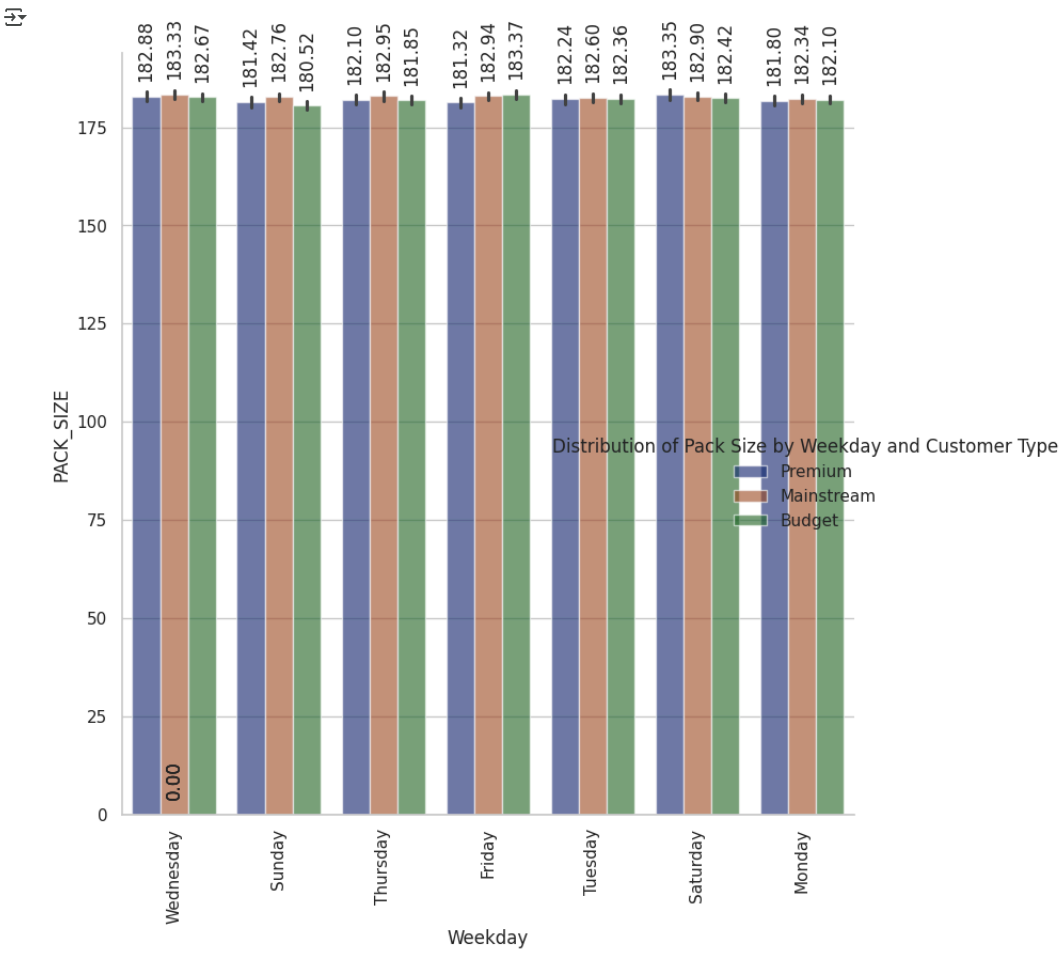

```
sns.set_theme(style="whitegrid")
```

```
g = sns.catplot(
    data=data, kind="bar",
    x="Weekday", y="TOT_SALES", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Weekday", "TOT_SALES")
g.legend.set_title("Distribution of Total Sales by Weekday and Customer Type")
```

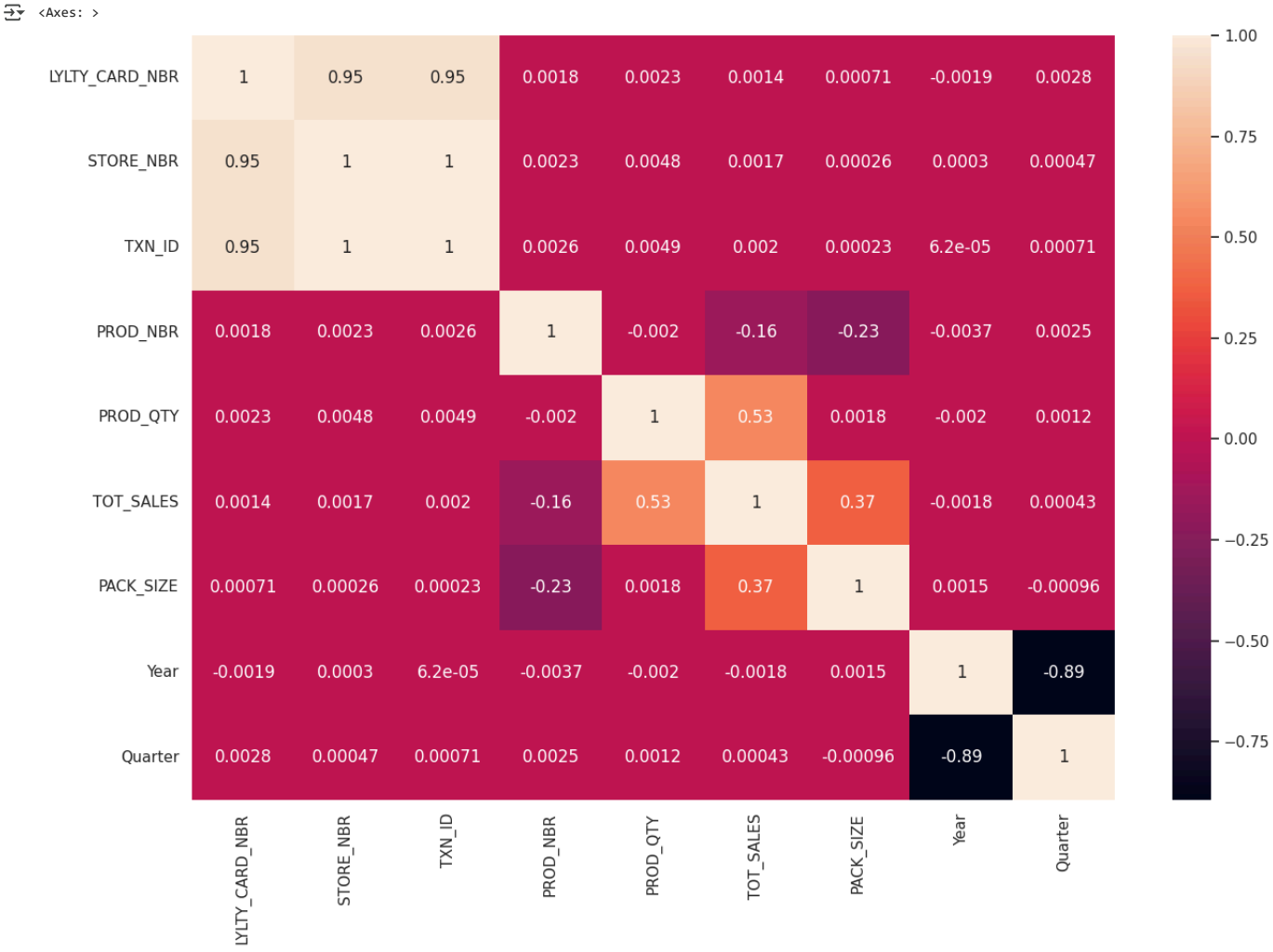
[1]




```
sns.set_theme(style="whitegrid")
g = sns.catplot(
    data=data, kind="bar",
    x="Weekday", y="PACK_SIZE", hue="PREMIUM_CUSTOMER", palette="dark", alpha=0.6, height=8, aspect=1.0
)


for ax in g.axes.flat:
    for p in ax.patches:
        ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
                    ha='center', va='bottom', xytext=(0, 10), textcoords='offset points', rotation=90)

g.set_xticklabels(rotation=90)
g.set_axis_labels("Weekday", "PACK_SIZE")
g.legend.set_title("Distribution of Pack Size by Weekday and Customer Type")
```




```
plt.figure(figsize=(15, 10))
# Select only numerical features for correlation calculation
numerical_data = data.select_dtypes(include=np.number)
sns.heatmap(numerical_data.corr(), annot=True)
```



3. Experimenting and Uplift Testing

```
total_sales=data.groupby(["STORE_NBR", "Month_Year"])["TOT_SALES"].sum()
total_sales=total_sales.to_frame()
```

```
total_sales.head(15)
```

STORE_NBR	Month_Year	TOT_SALES
1	April 2019	192.9
	August 2018	176.1
	December 2018	189.6
	February 2019	225.4
	January 2019	154.8
	July 2018	206.9
	June 2019	174.1
	March 2019	192.9
	May 2019	221.4
	November 2018	192.6
	October 2018	188.1
	September 2018	278.8
2	April 2019	196.5
	August 2018	193.8
	December 2018	136.0

Next steps: Generate code with total_sales | View recommended plots | New interactive sheet

```
total_customers=data.groupby(["STORE_NBR", "Month_Year"])["LYLTY_CARD_NBR"].nunique()
total_customers=total_customers.to_frame()
total_customers.head(15)
```

STORE_NBR	Month_Year	LYLTY_CARD_NBR
1	April 2019	42
	August 2018	42
	December 2018	42
	February 2019	52
	January 2019	35
	July 2018	49
	June 2019	42
	March 2019	45
	May 2019	46
	November 2018	46
	October 2018	44
	September 2018	59
2	April 2019	47
	August 2018	39
	December 2018	35

Next steps: Generate code with total_customers | View recommended plots | New interactive sheet

```
total_customers=data.groupby(["STORE_NBR", "Month_Year"])["LYLTY_CARD_NBR"].nunique()
total_customers=total_customers.to_frame()
total_customers.head(15)
```

STORE_NBR	Month_Year	LYLTY_CARD_NBR
1	April 2019	42
	August 2018	42
	December 2018	42
	February 2019	52
	January 2019	35
	July 2018	49
	June 2019	42
	March 2019	45
	May 2019	46
	November 2018	46
	October 2018	44
	September 2018	59
2	April 2019	47
	August 2018	39
	December 2018	35

Next steps: Generate code with total_customers | View recommended plots | New interactive sheet

```
transactions_per_customer=data.groupby(["STORE_NBR", "Month_Year"])["TXN_ID"].nunique()/data.groupby(["STORE_NBR", "Month_Year"])["LYLTY_CARD_NBR"].nunique()
#Grouping the pandas.DataFrame by the STORE_NBR and YEAR_MONTH column, and counting the unique number of transactions per customer for them.
transactions_per_customer=transactions_per_customer.to_frame()
transactions_per_customer.head(15)
```

		0
STORE_NBR	Month_Year	
1	April 2019	1.023810
	August 2018	1.023810
	December 2018	1.119048
	February 2019	1.057692
	January 2019	1.028571
	July 2018	1.061224
	June 2019	1.000000
	March 2019	1.088889
	May 2019	1.108696
	November 2018	1.021739
	October 2018	1.022727
	September 2018	1.050847
2	April 2019	1.042553
	August 2018	1.102564
	December 2018	1.057143

Next steps: Generate code with transactions_per_customer | View recommended plots | New interactive sheet

```
dataframe_list=[total_sales, total_customers, transactions_per_customer]
dataframe=pd.concat(dataframe_list, axis=1)
dataframe.columns=["TOT_SALES", "TOT_CUST", "TXN_PER_CUST"]
dataframe.head(15)
```

		TOT_SALES	TOT_CUST	TXN_PER_CUST
STORE_NBR	Month_Year			
1	April 2019	192.9	42	1.023810
	August 2018	176.1	42	1.023810
	December 2018	189.6	42	1.119048
	February 2019	225.4	52	1.057692
	January 2019	154.8	35	1.028571
	July 2018	206.9	49	1.061224
	June 2019	174.1	42	1.000000
	March 2019	192.9	45	1.088889
	May 2019	221.4	46	1.108696
	November 2018	192.6	46	1.021739
	October 2018	188.1	44	1.022727
	September 2018	278.8	59	1.050847
2	April 2019	196.5	47	1.042553
	August 2018	193.8	39	1.102564
	December 2018	136.0	35	1.057143

Next steps: Generate code with dataframe | View recommended plots | New interactive sheet

```
dataframe.isnull().sum()
```

	0
TOT_SALES	0
TOT_CUST	0
TXN_PER_CUST	0

dtype: int64

```
recorded_stores=pd.pivot_table(data, index="STORE_NBR", columns="Month_Year", values="TXN_ID", aggfunc="count")
recorded_stores
```

Month_Year	April 2019	August 2018	December 2018	February 2019	January 2019	July 2018	June 2019	March 2019	May 2019	November 2018	October 2018	September 2018
STORE_NBR												
1	43.0	43.0	47.0	55.0	36.0	52.0	43.0	49.0	51.0	47.0	45.0	62.0
2	49.0	43.0	38.0	32.0	45.0	41.0	42.0	46.0	50.0	40.0	43.0	37.0
3	110.0	134.0	129.0	139.0	121.0	138.0	122.0	130.0	123.0	118.0	119.0	119.0
4	137.0	151.0	133.0	102.0	168.0	160.0	134.0	135.0	126.0	139.0	155.0	138.0
5	109.0	112.0	125.0	106.0	118.0	120.0	127.0	97.0	104.0	111.0	107.0	125.0
...	...	...	...	...	...	...	...	...	...	...	...	...
268	50.0	54.0	43.0	37.0	38.0	52.0	40.0	47.0	52.0	51.0	48.0	34.0
269	139.0	132.0	133.0	133.0	144.0	139.0	127.0	122.0	130.0	136.0	148.0	124.0
270	132.0	154.0	149.0	125.0	155.0	139.0	127.0	143.0	128.0	133.0	119.0	126.0
271	109.0	101.0	117.0	102.0	120.0	129.0	129.0	101.0	127.0	122.0	114.0	114.0
272	56.0	48.0	47.0	48.0	50.0	52.0	37.0	53.0	40.0	45.0	51.0	36.0

272 rows × 12 columns

Next steps: Generate code with recorded_stores | View recommended plots | New interactive sheet

```
recorded_stores.isnull().sum()
```

Month_Year	0
April 2019	7
August 2018	9
December 2018	9
February 2019	8
January 2019	9
July 2018	6
June 2019	8
March 2019	7
May 2019	9
November 2018	8
October 2018	7
September 2018	8

dtype: int64

```
unrecorded_stores=[]
for i in recorded_stores.index:
    if recorded_stores.loc[i].isnull().any():
        unrecorded_stores.append(i)
unrecorded_stores
```

```
[11, 31, 44, 76, 85, 92, 117, 193, 206, 211, 218, 252]
```

```
dataframe=dataframe.drop(unrecorded_stores, axis=0)
dataframe
```

STORE_NBR	Month_Year	TOT_SALES	TOT_CUST	TXN_PER_CUST
1	April 2019	192.9	42	1.023810
	August 2018	176.1	42	1.023810
	December 2018	189.6	42	1.119048
	February 2019	225.4	52	1.057692
	January 2019	154.8	35	1.028571
...	...	...	...	...
272	March 2019	442.3	50	1.060000
	May 2019	314.6	34	1.176471
	November 2018	376.2	41	1.097561
	October 2018	430.6	44	1.136364
	September 2018	304.7	32	1.125000

3120 rows × 3 columns

Next steps: Generate code with dataframe | View recommended plots | New interactive sheet

A. Pre-Trial Duration -- Before February 2019

```
pre_trial_data = dataframe.loc[pd.to_datetime(dataframe.index.get_level_values("Month_Year"), format="%B %Y") < "2019-02"]
```

```
pre_trial_data=pre_trial_data.reset_index()
```

```
pre_trial_data
```

	STORE_NBR	Month_Year	TOT_SALES	TOT_CUST	TXN_PER_CUST
0	1	August 2018	176.1	42	1.023810
1	1	December 2018	189.6	42	1.119048
2	1	January 2019	154.8	35	1.028571
3	1	July 2018	206.9	49	1.061224
4	1	November 2018	192.6	46	1.021739
...	...	...	...	...	...
1815	272	January 2019	423.0	46	1.086957
1816	272	July 2018	433.1	48	1.083333
1817	272	November 2018	376.2	41	1.097561
1818	272	October 2018	430.6	44	1.136364
1819	272	September 2018	304.7	32	1.125000

1820 rows × 5 columns

Next steps: Generate code with pre_trial_data | View recommended plots | New interactive sheet

```
control_stores=pre_trial_data[(pre_trial_data.STORE_NBR!=77 ) & (pre_trial_data.STORE_NBR!=86) & (pre_trial_data.STORE_NBR!=88)][["TOT_SALES", "TOT_CUST", "TXN_PER_CUST"]].groupby(pre_trial_
control_stores
```

STORE_NBR	TOT_SALES	TOT_CUST	TXN_PER_CUST
1	1386.90	317	7.327967
2	1128.50	272	7.359700
3	7526.15	744	8.209829
4	9127.00	849	8.535253
5	5739.70	651	8.791906
...	...	...	...
268	1549.05	304	7.373037
269	6664.50	746	8.921035
270	6697.95	734	9.147187
271	5765.10	652	8.671966
272	2744.35	302	7.620124

Next steps: Generate code with control_stores | View recommended plots | New interactive sheet

257 rows × 3 columns

```
trial_stores=pre_trial_data[(pre_trial_data.STORE_NBR==77 ) | (pre_trial_data.STORE_NBR==86) | (pre_trial_data.STORE_NBR==88)][["TOT_SALES", "TOT_CUST", "TXN_PER_CUST"]].groupby(pre_trial_da
trial_stores
```

STORE_NBR	TOT_SALES	TOT_CUST	TXN_PER_CUST
77	1699.00	299	7.405289
86	6119.85	697	8.798544
88	9383.60	880	8.523817

Next steps: Generate code with trial_stores | View recommended plots | New interactive sheet

Store Number : 77

```
difference=control_stores.loc[control_stores.corrwith(trial_stores.loc[77], method="pearson", axis=1).nlargest(5).index]

difference=(trial_stores.loc[77]-difference).sort_values(by="TOT_SALES", ascending=False)
difference["DIFFERENCE"]=difference["TOT_SALES"]-difference["TOT_SALES"].mean()
difference.sort_values(by="DIFFERENCE", ascending=False)
```

STORE_NBR	TOT_SALES	TOT_CUST	TXN_PER_CUST	DIFFERENCE
139	1493.2	257.0	0.405289	609.34
135	1486.9	256.0	0.012432	603.04
161	1459.0	252.0	0.405289	575.14
233	39.2	1.0	0.115969	-844.66
46	-59.0	-3.0	0.094215	-942.86

Store Number : 86

```
difference=control_stores.loc[control_stores.corrwith(trial_stores.loc[86], axis=1).nlargest(5).index]
difference=(trial_stores.loc[86]-difference).sort_values(by="TOT_SALES", ascending=False)
difference["DIFFERENCE"]=difference["TOT_SALES"]-difference["TOT_SALES"].mean()
difference.sort_values(by="DIFFERENCE", ascending=False)
```

STORE_NBR	TOT_SALES	TOT_CUST	TXN_PER_CUST	DIFFERENCE
258	5934.85	670.0	1.798544	4066.46
215	3411.85	386.0	1.486773	1543.46
225	29.25	3.0	0.023669	-1839.14