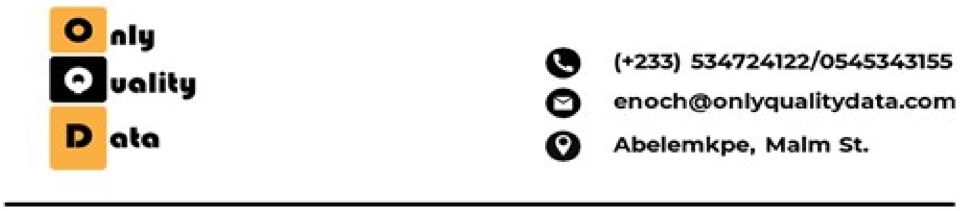
QUANTIUM DATA ANALYTICS PROJECT REPORT

Authored by Abrokwah Joseph Junior

In [10]:
```
from IPython.display import Image
h=600
w=600
Image(filename=r"C:\Users\user\Desktop\OQD Internship\OQD Leterhead.jpg", height
```

Out[10]:



Introduction

This Report is an in-depth analysis of **Customer and Transaction Behaviour** of a fictional chips company. As part of a **Data Analytics Internship** hosted by **ONLY QUALITY DATA**, two datasets **Q_Purchase_Behaviour** and **Q_Transaction_Data** obtained from **Forage**, under the *Quantium Data Analytics job simulation*, were used for the analyses.

Objective of the Project

The project's objective was to enable interns to sharpen their data analytics skills by using __Python__ as a Data Analytics Tool.

Tools Used for the Analyses

- **Pandas** for data handling and analyses
- **Plotly.Express**, **Matplotlib** and **Seaborn** for data visualizations

About the Datasets

The nature of the various columns in the datasets is described below

Data Dictionary: Q_Transaction_Data Dataset

Column Name	Data Type	Description
DATE	Numeric	The date of the transaction, represented as a numeric format. Could be converted to an actual date.
STORE_NBR	Integer	The unique identifier for the store where the transaction took place.
LYLTY_CARD_NBR	Integer	The loyalty card number of the customer, representing a unique customer identifier.
TXN_ID	Integer	The unique transaction ID for each purchase.
PROD_NBR	Integer	The unique product number identifying the specific product purchased.
PROD_NAME	String	The name of the product purchased, including the product description and pack size.
PROD_QTY	Integer	The quantity of the product purchased in the transaction.
TOT_SALES	Numeric	The total sales value in dollars for the specific transaction and product.

Data Dictionary: Q_Purchase_Behaviour Dataset

Column Name	Data Type	Description	Distinct Categories
LIFESTAGE	Categorical	Represents the life stage of the customer, helping to segment the customer base by their family situation and age group.	- MIDGE SINGLES/COUPLES - NEW FAMILIES - OLDER FAMILIES - OLDER SINGLES/COUPLES - RETIREES - YOUNG FAMILIES - YOUNG SINGLES/COUPLES
PREMIUM_CUSTOMER	Categorical	Represents the financial status or purchasing power of the customer, indicating their spending behavior.	- Budget - Mainstream - Premium

Data Cleaning and Preparation

- The datasets, of which both were Microsoft Excel documents, were imported into the notebook to enable working on them to be worked on.
- Each of the columns in the datasets was then converted to their standard data types.
- the columns were examined for missing data and outliers (data points with extremely high or low values). The outliers were then removed.
- The two datasets were merged to form one 'big' dataset which was named *merged_df*
- Relevant extra columns, 'UNIT PRICE' (the unit prices of the various products) and 'MONTH-YEAR' (month and year) were created from existing columns
- The numerical columns were aggregated as sums and means according to the relevant categories
- Different charts were generated to visualize the stories the data tells

In [4]:
```
# Import libraries

import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
```

Dataset importation and inspection

In [5]:
```
# Load Purchase Behavior dataset
behavior_df=pd.read_excel('Q_Purchase_Behaviour.xlsx')

# Load Transaction Dataset
transaction_df=pd.read_excel('Q_Transaction_Data.xlsx')
```

In [6]:
```
behavior_df.head()
```

Out[6]:

	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

In [11]:
```
transaction_df.head()
```

Out[11]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_Q
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	

In [12]:
```
behavior_df.info()
```

```
<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
```

In [13]:
```
transaction_df.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  datetime64[ns]
 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: datetime64[ns](1), float64(1), int64(5), object(1)
memory usage: 16.2+ MB
```

In [14]:
```
#changing data types in the transaction table
transaction_df[['STORE_NBR', 'LYLTY_CARD_NBR','TXN_ID','PROD_NBR']]=transaction_
```

In [15]:
```
transaction_df.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  datetime64[ns]
 1   STORE_NBR       264836 non-null  object        
 2   LYLTY_CARD_NBR  264836 non-null  object        
 3   TXN_ID          264836 non-null  object        
 4   PROD_NBR        264836 non-null  object        
 5   PROD_NAME       264836 non-null  object        
 6   PROD_QTY        264836 non-null  int64         
 7   TOT_SALES       264836 non-null  float64       
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 16.2+ MB
```

In [16]:
```
behavior_df['LYLTY_CARD_NBR']=behavior_df['LYLTY_CARD_NBR'].astype(str)
```

In [17]:
```
behavior_df.info()
```

```
<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  object
 1   LIFESTAGE         72637 non-null  object
 2   PREMIUM_CUSTOMER  72637 non-null  object
dtypes: object(3)
memory usage: 1.7+ MB
```

In [18]:
```
#search for missing values
print(behavior_df.isna().sum())
print('')
print(transaction_df.isna().sum())
```

```
LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64

DATE              0
STORE_NBR         0
LYLTY_CARD_NBR    0
TXN_ID            0
PROD_NBR          0
PROD_NAME         0
PROD_QTY          0
TOT_SALES         0
dtype: int64
```

In [19]:
```
# merging dataframes using Loyalty Card Number as reference
merge_df= transaction_df.merge(behavior_df, on='LYLTY_CARD_NBR')
```

In [20]:
```
merge_df.head()
```

Out[20]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_Q
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	

In [21]:
```
# Create Unit Price column from total sales and product quantity columns
merge_df['UNIT PRICE']= merge_df['TOT_SALES']/merge_df['PROD_QTY']
```

In [22]:
```
# Creat a month-year column from date column
merge_df['MONTH-YEAR']=merge_df['DATE'].dt.to_period('M').dt.to_timestamp()
```

In [23]:
```
merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264836 entries, 0 to 264835
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   DATE              264836 non-null  datetime64[ns]
 1   STORE_NBR         264836 non-null  object
 2   LYLTY_CARD_NBR    264836 non-null  object
 3   TXN_ID            264836 non-null  object
 4   PROD_NBR          264836 non-null  object
 5   PROD_NAME         264836 non-null  object
 6   PROD_QTY          264836 non-null  int64
 7   TOT_SALES         264836 non-null  float64
 8   LIFESTAGE         264836 non-null  object
 9   PREMIUM_CUSTOMER  264836 non-null  object
 10  UNIT PRICE        264836 non-null  float64
 11  MONTH-YEAR        264836 non-null  datetime64[ns]
dtypes: datetime64[ns](2), float64(2), int64(1), object(7)
memory usage: 24.2+ MB
```

In [24]:
```
merge_df.describe()
```

Out[24]:

	DATE	PROD_QTY	TOT_SALES	UNIT PRICE	MONTH-YE
count	264836	264836.000000	264836.000000	264836.000000	2648
mean	2018-12-30 00:52:12.879215616	1.907309	7.304200	3.824624	2018-12 07:45:14.368136
min	2018-07-01 00:00:00	1.000000	1.500000	1.320000	2018-07 00:00
25%	2018-09-30 00:00:00	2.000000	5.400000	3.000000	2018-09 00:00
50%	2018-12-30 00:00:00	2.000000	7.400000	3.800000	2018-12 00:00
75%	2019-03-31 00:00:00	2.000000	9.200000	4.600000	2019-03 00:00
max	2019-06-30 00:00:00	200.000000	650.000000	6.500000	2019-06 00:00
std	NaN	0.643654	3.083226	1.109523	N

In [25]:
```
 #check for missing values
merge_df.isna().sum()
```

Out[25]:
```
DATE                0
STORE_NBR           0
LYLTY_CARD_NBR      0
TXN_ID              0
PROD_NBR            0
PROD_NAME           0
PROD_QTY            0
TOT_SALES           0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
UNIT PRICE          0
MONTH-YEAR          0
dtype: int64
```

In [26]:
```
len(merge_df)
```

Out[26]: 264836

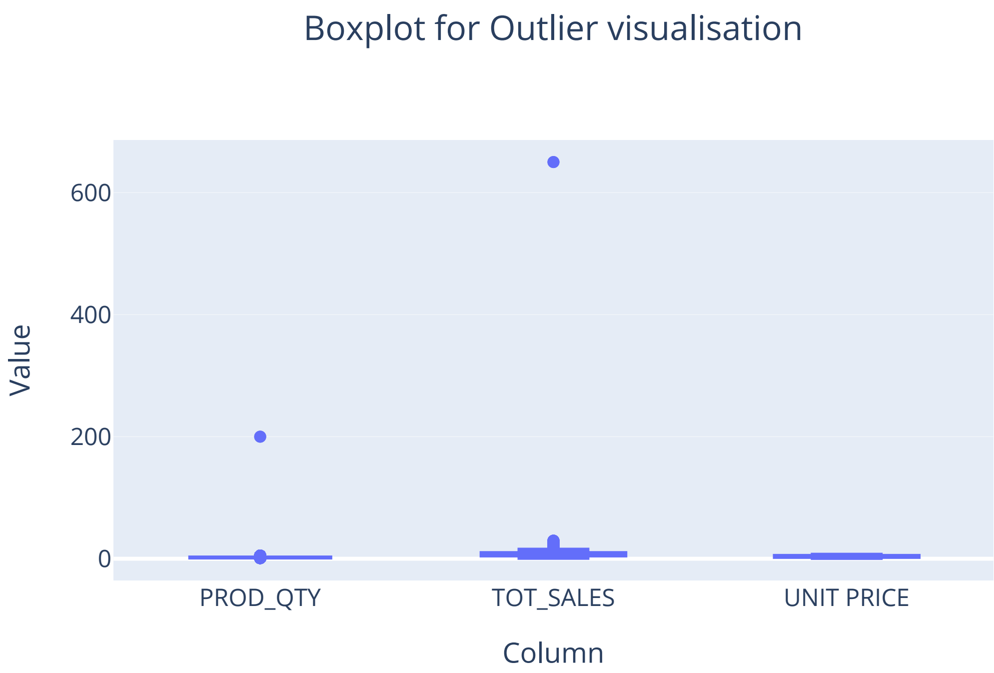
In [27]:
```
merge_df.head()
```

Out[27]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_Q
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	

Looking for Outliers

In [28]:
```
#Looking for outliers
num_columns=merge_df[["PROD_QTY", 'TOT_SALES','UNIT PRICE']] # Extract numerical
reshape_df=num_columns.melt(var_name='Column',value_name='Value') # Reshape data
fig=px.box(reshape_df,
           x='Column', y='Value',
          title= 'Boxplot for Outlier visualisation',
          orientation="v")
fig.update_layout(width=600, height=400, title_x=0.5)
fig.show()
```



In [29]:
```
# sort data in descending order based on product quantity to reveal outliers
merge_df.sort_values(by='PROD_QTY', ascending=False).head(10)
```

Out[29]:

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PRO
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	
217237	2019-05-18	201	201060	200202	26	Pringles Sweet&Spcy BBQ 134g	
238333	2018-08-14	219	219004	218018	25	Pringles SourCream Onion 134g	
238471	2019-05-19	261	261331	261111	87	Infuzions BBQ Rib Prawn Crackers 110g	
228749	2019-05-19	232	232138	235978	109	Pringles Barbeque 134g	
117802	2019-05-19	176	176471	177469	17	Kettle Sensations BBQ&Maple 150g	
228711	2018-08-17	205	205149	204215	1	Smiths Crinkle Cut Chips Barbecue 170g	
238397	2019-05-18	238	238337	243243	28	Thins Potato Chips Hot & Spicy 175g	
238395	2019-05-19	238	238250	242874	88	Kettle Honey Soy Chicken 175g	

In [30]:

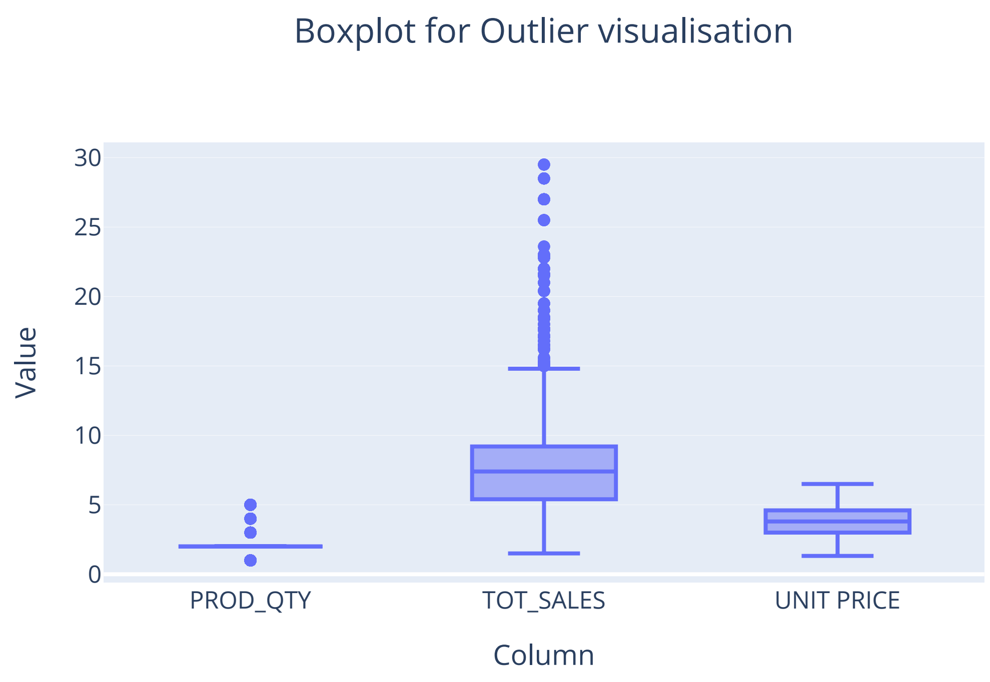
```
#subsetting outliers
merge_df[merge_df["PROD_QTY"]>150]
```

Out[30]:

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

In [31]:
```
#removing outliers
merge_df.drop([69762,69763], inplace=True)
```

In [32]:
```
#Looking for outliers after removing datapoints with extreme values
numerical_columns=merge_df[["PROD_QTY", 'TOT_SALES','UNIT PRICE']] # Extract num
reshaped_df=numerical_columns.melt(var_name='Column',value_name='Value') # Resha
fig=px.box(reshaped_df,
           x='Column', y='Value',
          title= 'Boxplot for Outlier visualisation',
          orientation="v")
fig.update_layout(width=600, height=400, title_x=0.5)
fig.show()
```



Calculated key mearsures

In [33]:
```
#calculating measures
print(f"Total Sales: ${merge_df['TOT_SALES'].sum():.2f}")
print("")
print(f"Total Quantity sold: {merge_df['PROD_QTY'].sum()}")
print("")
```

```
print(f"Number of Customers: {merge_df['LYLTY_CARD_NBR'].nunique()}")
print("")
print(f"Total Number of  Transactions: {transaction_df['TXN_ID'].count()}")
```

```
Total Sales: $1933115.00

Total Quantity sold: 504724

Number of Customers: 72636

Total Number of  Transactions: 264836
```

Aggregating data

In [34]:
```
merge_df.columns
```

Out[34]:
```
Index(['DATE', 'STORE_NBR', 'LYLTY_CARD_NBR', 'TXN_ID', 'PROD_NBR',
       'PROD_NAME', 'PROD_QTY', 'TOT_SALES', 'LIFESTAGE', 'PREMIUM_CUSTOMER',
       'UNIT PRICE', 'MONTH-YEAR'],
      dtype='object')
```

In [35]:
```
# calculate total sale for each lifestage category
lifestage_by_sales=merge_df.groupby("LIFESTAGE")["TOT_SALES"].sum().reset_index(

# calculate total sales made by each store and rank them in descending order
top_stores_by_sales= merge_df.groupby("STORE_NBR")['TOT_SALES'].sum().reset_inde

# calculate total quantity of products sold by each store and rank them in desce
top_stores_by_product_qty= merge_df.groupby("STORE_NBR")['PROD_QTY'].sum().reset

# calculate total sales made from each product and rank them in descending order
top_products_by_sales= merge_df.groupby("PROD_NAME")['TOT_SALES'].sum().reset_in

# calculate total quantity of each product and rank them in descending order
top_products_by_product_qty= merge_df.groupby("PROD_NAME")['PROD_QTY'].sum().res

# calculate the price of each product and rank them in descending order
top_products_by_unit_price= merge_df.groupby("PROD_NAME")['UNIT PRICE'].mean().r

#count number of members in the customer groups
premuim_customer=behavior_df['PREMIUM_CUSTOMER'].value_counts().reset_index().re

# calculate total sales made from each customer and rank them in descending orde
most_active_customers= merge_df.groupby('LYLTY_CARD_NBR')['TOT_SALES'].sum().res
```

In [36]:
```
lifestage_by_sales
```

Out[36]:

	LIFESTAGE	TOT_SALES
3	OLDER SINGLES/COUPLES	402426.75
4	RETIREES	366470.90
2	OLDER FAMILIES	352467.20
5	YOUNG FAMILIES	316160.10
6	YOUNG SINGLES/COUPLES	260405.30
0	MIDAGE SINGLES/COUPLES	184751.30
1	NEW FAMILIES	50433.45

In [37]:
```
top_stores_by_sales.head()
```

Out[37]:

	STORE_NBR	TOT_SALES
141	226	17605.45
259	88	16333.25
73	165	15973.75
207	40	15559.50
153	237	15539.50

In [38]:
```
top_stores_by_product_qty.head()
```

Out[38]:

	STORE_NBR	PROD_QTY
141	226	4001
259	88	3718
265	93	3639
73	165	3602
210	43	3519

In [39]:
```
top_products_by_sales.head()
```

Out[39]:

	PROD_NAME	TOT_SALES
11	Dorito Corn Chp Supreme 380g	39052.0
86	Smiths Crnkle Chip Orgnl Big Bag 380g	36367.6
77	Smiths Crinkle Chips Salt & Vinegar 330g	34804.2
33	Kettle Mozzarella Basil & Pesto 175g	34457.4
76	Smiths Crinkle Original 330g	34302.6

In [40]:
```
top_products_by_product_qty.head()
```

Out[40]:

	PROD_NAME	PROD_QTY
33	Kettle Mozzarella Basil & Pesto 175g	6381
42	Kettle Tortilla ChpsHny&Jlpno Chili 150g	6309
8	Cobs Popd Sea Salt Chips 110g	6277
10	Cobs Popd Swt/Chlli &Sr/Cream Chips 110g	6256
98	Tostitos Splash Of Lime 175g	6234

In [41]:
```
top_products_by_unit_price.head(5)
```

Out[41]:

	PROD_NAME	UNIT PRICE
11	Dorito Corn Chp Supreme 380g	6.368285
86	Smiths Crnkle Chip Orgnl Big Bag 380g	5.900000
12	Doritos Cheese Supreme 330g	5.700000
76	Smiths Crinkle Original 330g	5.700000
77	Smiths Crinkle Chips Salt & Vinegar 330g	5.700000

In [42]:
```
most_active_customers.head()
```

Out[42]:

	LYLTY_CARD_NBR	TOT_SALES
37566	230078	138.6
61529	63197	132.8
46591	259009	127.2
17124	162039	126.8
60028	58361	124.8

In [43]:
```
premuim_customer
```

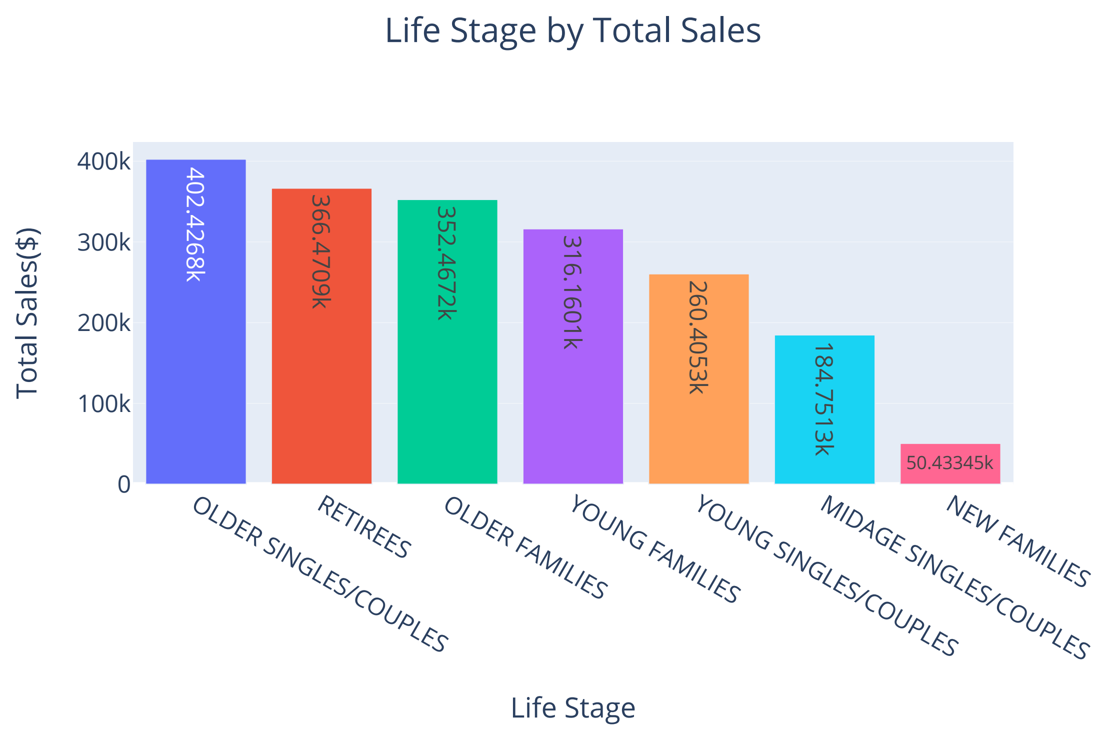
Out[43]:

	PREMIUM_CUSTOMER	No. of Customers
0	Mainstream	29245
1	Budget	24470
2	Premium	18922

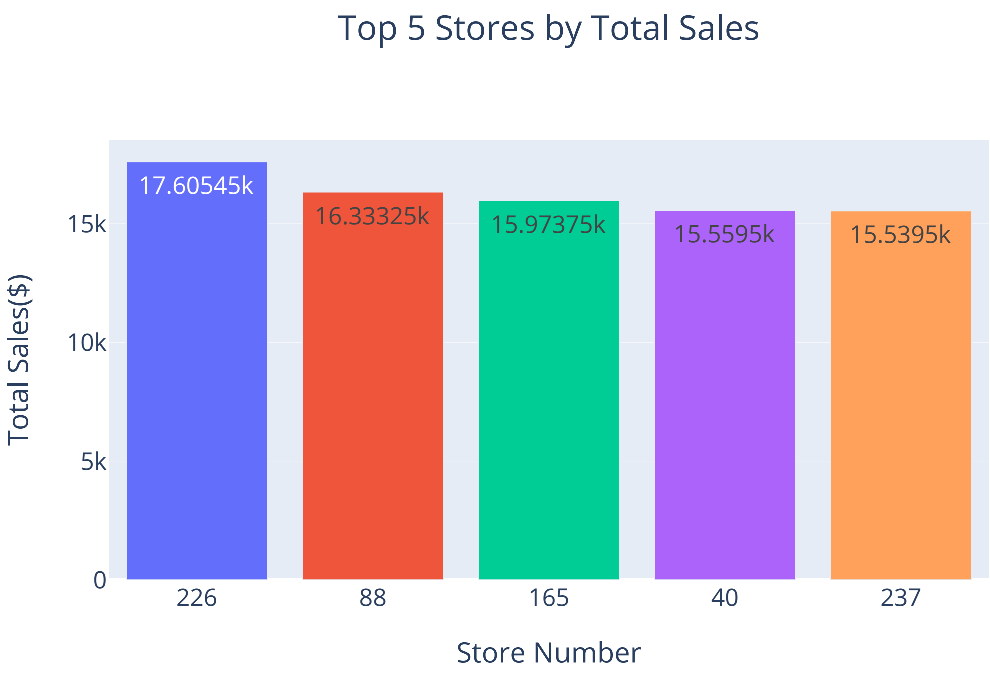
Visualisation

In [44]:
```
# visualizing tottal sales across lifestages
fig=px.bar(lifestage_by_sales.sort_values(by='TOT_SALES', ascending=False),
           x="LIFESTAGE",y="TOT_SALES",
          title='Life Stage by Total Sales',
          text_auto=True,
          color="LIFESTAGE",
```

```
          labels={"LIFESTAGE":'Life Stage', "TOT_SALES": "Total Sales($)"})
fig.update_layout(xaxis_title="Life Stage",
                  yaxis_title="Total Sales($)",
                  width=600, height=400,
                  title_x=0.5,
                  showlegend=False)
fig.show()
```



In [45]:
```
# visualizing top 5 stores that made the most sales
fig=px.bar(top_stores_by_sales.head(), x="STORE_NBR",y="TOT_SALES",
          title='Top 5 Stores by Total Sales',
          color="STORE_NBR",
          labels={"STORE_NBR": 'Store NO.', "TOT_SALES":"Total Sales($)"},
          text_auto=True)
fig.update_layout(xaxis_title="Store Number",
                  yaxis_title="Total Sales($)", width=600, height=400,
                  title_x=0.5,
                  showlegend=False
                 )
fig.show()
```



In [46]:
```
# visualizing top 5 stores that sold most products
fig=px.bar(top_stores_by_product_qty.head(), x="STORE_NBR",y="PROD_QTY",
          title='Top 5 Stores by Product Quantity',
          text_auto=True,
          color="STORE_NBR",
          labels={"STORE_NBR":'Store NO.', "PROD_QTY":"Product Quantity Sold"})
fig.update_layout(xaxis_title= "Store Number",yaxis_title="Product Quantity Sold
                  width=600, height=400,title_x=0.5,
                  showlegend=False
                 )
fig.show()
```

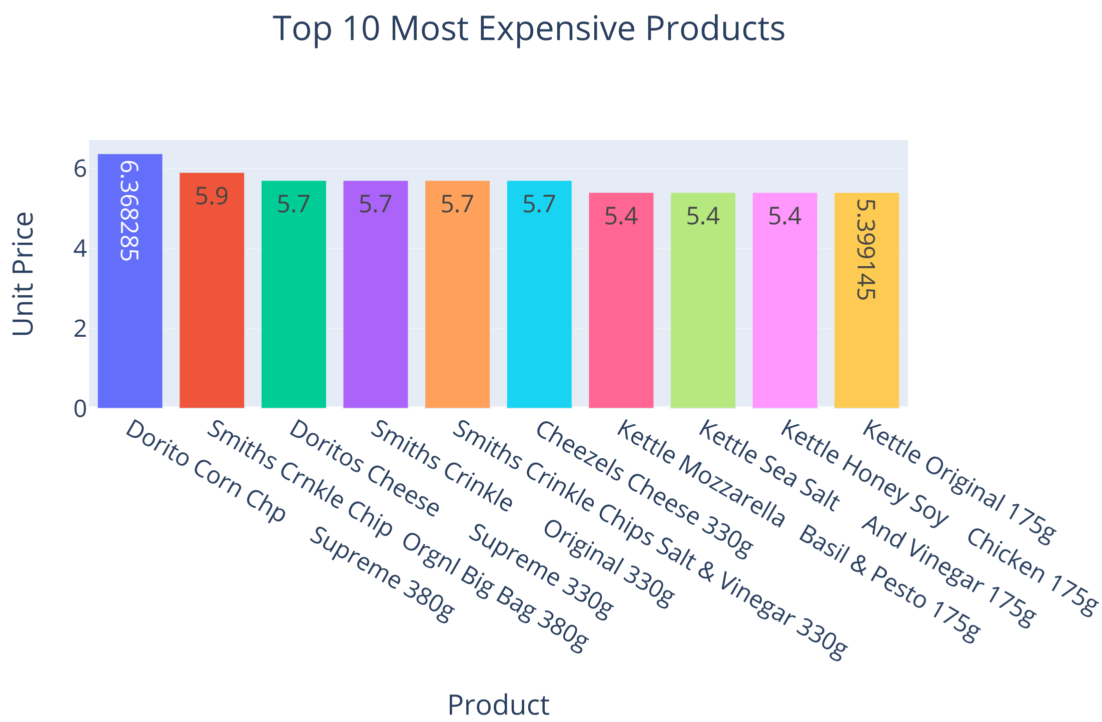
In [47]:
```python
# visualizing top 5 products that generated the most sales
fig=px.bar(top_products_by_sales.head(), x="PROD_NAME",y="TOT_SALES",
          title='Top 5 Products by Total Sales',
          text_auto=True,
          color="PROD_NAME",
          labels={"PROD_NAME":'Product', "TOT_SALES":"Total Sales ($)"})
fig.update_layout(xaxis_title="Product" ,yaxis_title="Total Sales ($)",
                  width=600, height=400,title_x=0.5,
                  showlegend=False
                 )
fig.show()
```

In [48]:

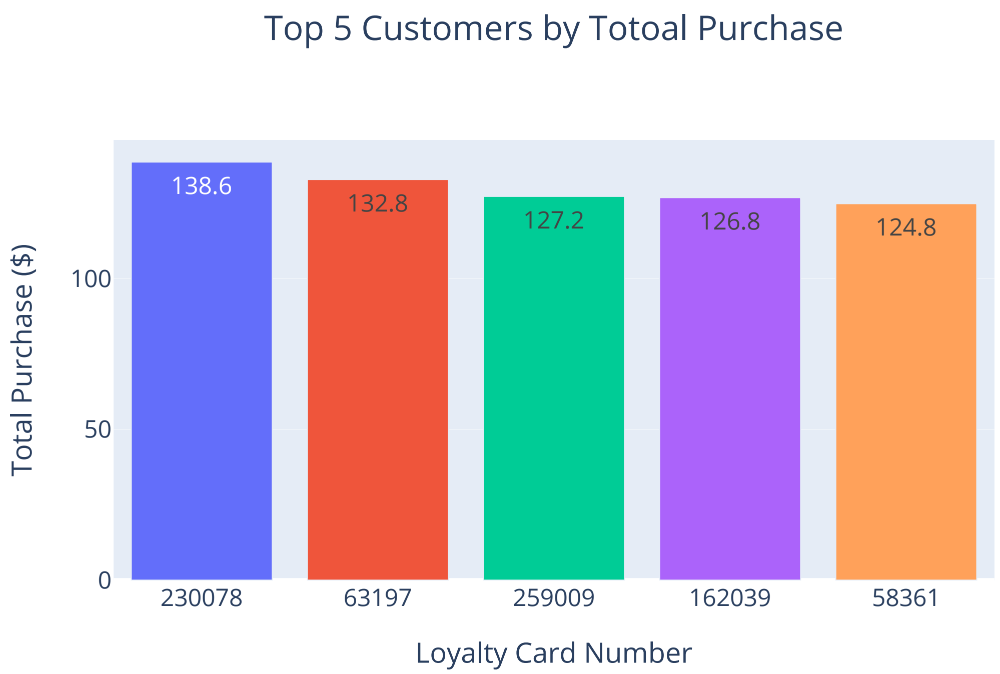
```
# visualizing top 5 products that were ordered most
fig=px.bar(top_products_by_product_qty.head(), x="PROD_NAME",y="PROD_QTY",
          title='Top 5 Products by Quantity Sold',
          text_auto=True,
          color="PROD_NAME",
          labels={"PROD_NAME":'Product', "PROD_QTY":"Quantity Sold"})
fig.update_layout(xaxis_title="Product" ,yaxis_title="Quantity Sold",
                  width=600, height=400,title_x=0.5,
                  showlegend=False
                 )
fig.show()
```

In [49]:

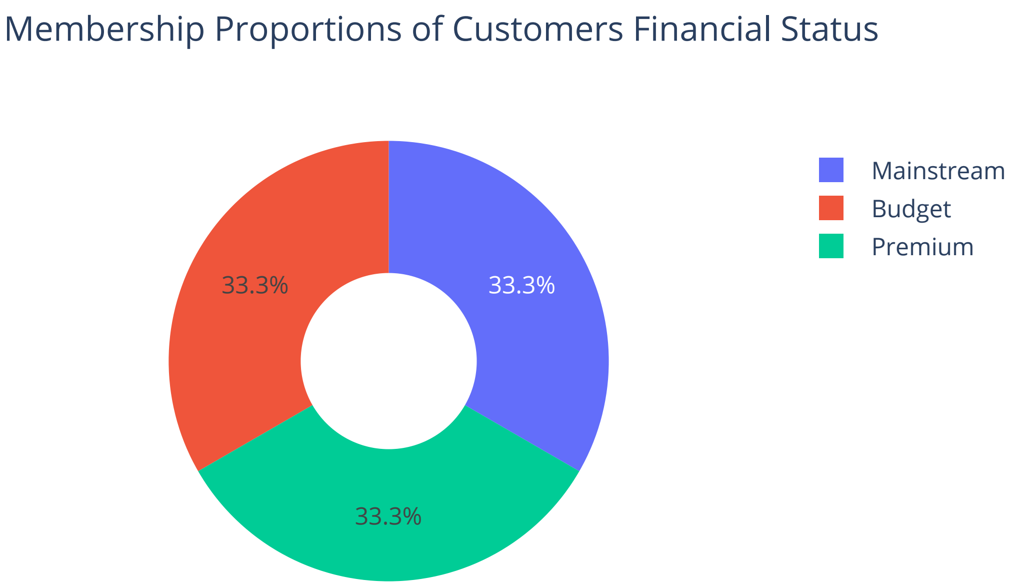
```
# visualizing top 10 most expensive products
fig=px.bar(top_products_by_unit_price.head(10), x="PROD_NAME",y="UNIT PRICE",
          title='Top 10 Most Expensive Products',
          text_auto=True,
          color="PROD_NAME",
          labels={"PROD_NAME":'Product','UNIT PRICE':"Price ($)"})
fig.update_layout(xaxis_title="Product" ,yaxis_title="Unit Price",
                   width=600, height=400,title_x=0.5,
                  showlegend=False
                 )
fig.show()
```



In [50]:
```
# visualizing top 5 performing customers
fig=px.bar(most_active_customers.head(), x='LYLTY_CARD_NBR',y='TOT_SALES',
          title='Top 5 Customers by Totoal Purchase',
          text_auto=True,
          color='LYLTY_CARD_NBR',
          labels={'LYLTY_CARD_NBR':'Loyalty Card Number','TOT_SALES':"Total Purc
fig.update_layout(xaxis_title="Loyalty Card Number" ,yaxis_title="Total Purchase
                  width=600, height=400,title_x=0.5,
                  showlegend=False
                 )
fig.show()
```




```
In [51]: #Visualize the premium customer category
fig=px.pie(premuim_customer, names='PREMIUM_CUSTOMER',title="Membership Proporti
           hover_data='No. of Customers',hole=0.4,
           labels={'PREMIUM_CUSTOMER':'Financial Status'},
           color='PREMIUM_CUSTOMER')
fig.update_layout(width=600, height=400, title_x=0.5)
fig.show()
```



Corelation Analysis

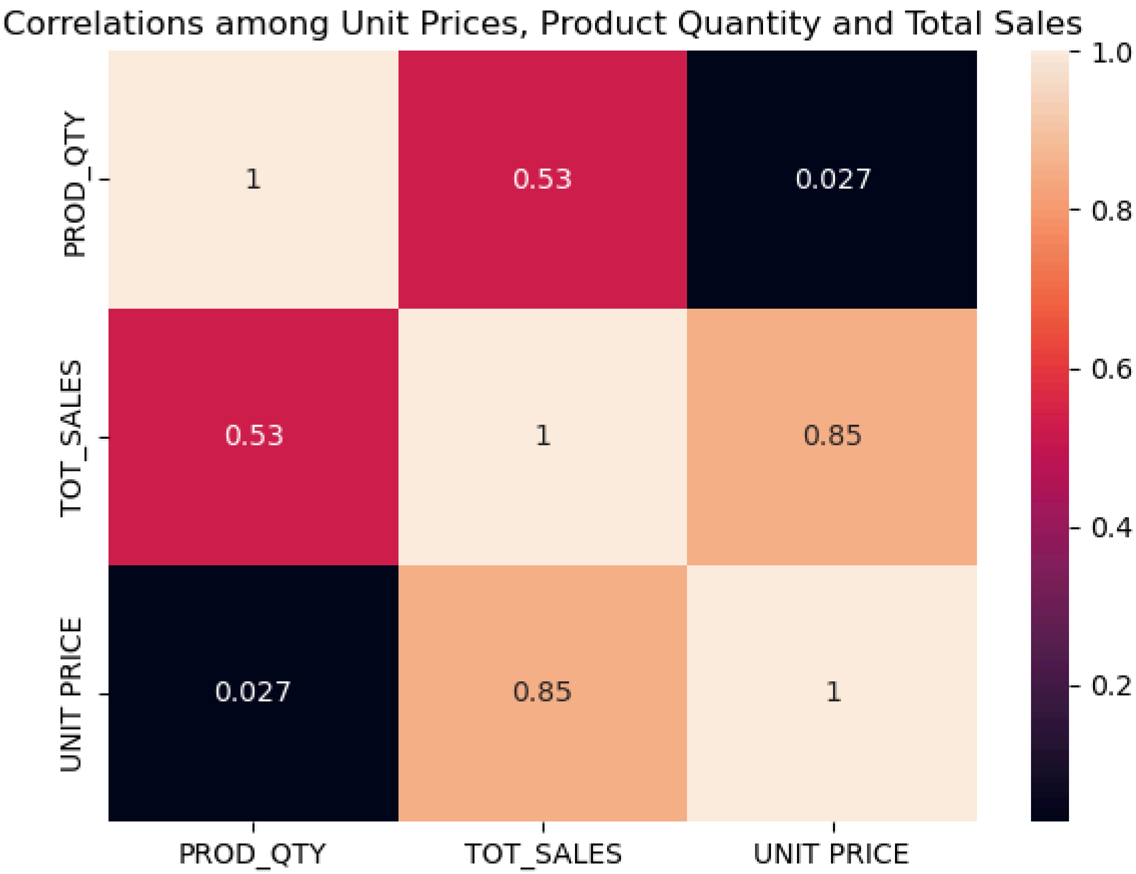
In [52]:
```
# Correlation coeffitients among Unit Prices, Product Quantity and Total Sales
corelations=merge_df[['PROD_QTY', 'TOT_SALES', "UNIT PRICE"]].corr()
corelations
```

Out[52]:

	PROD_QTY	TOT_SALES	UNIT PRICE
PROD_QTY	1.000000	0.527788	0.027083
TOT_SALES	0.527788	1.000000	0.849723
UNIT PRICE	0.027083	0.849723	1.000000

In [53]:
```
#Visualize Correlations among Unit Prices, Product Quantity and Total Sales
plt.figure(figsize=(7,5))
sns.heatmap(corelations, annot=True)
plt.title("Correlations among Unit Prices, Product Quantity and Total Sales")
;
```

Out[53]: ''



Date based analysis

In [54]:
```
# Aggregate data by date
date_trend=merge_df.groupby('DATE')[['PROD_QTY', 'TOT_SALES','UNIT PRICE']].sum(
date_trend.head()
```

Out[54]:

	DATE	PROD_QTY	TOT_SALES	UNIT PRICE
0	2018-07-01	1394	5372.2	2784.9
1	2018-07-02	1367	5315.4	2764.5
2	2018-07-03	1389	5321.8	2763.5
3	2018-07-04	1373	5309.9	2755.5
4	2018-07-05	1358	5080.9	2650.6

In [55]:

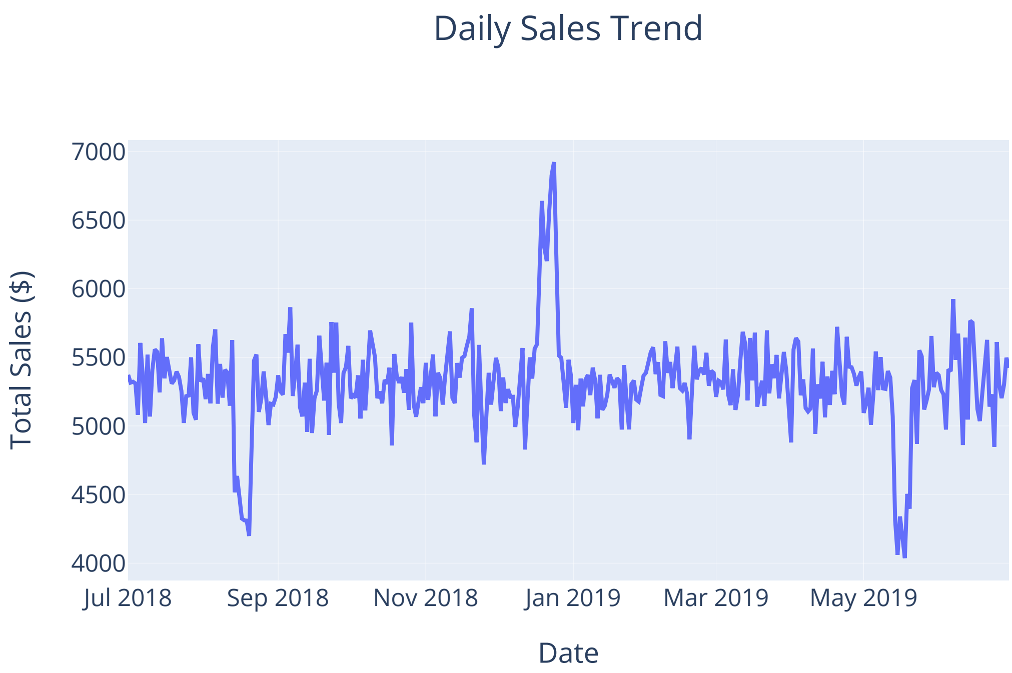
```
# Aggregate data by month
month_trend=merge_df.groupby('MONTH-YEAR')[['PROD_QTY', 'TOT_SALES','UNIT PRICE'
month_trend
```

Out[55]:

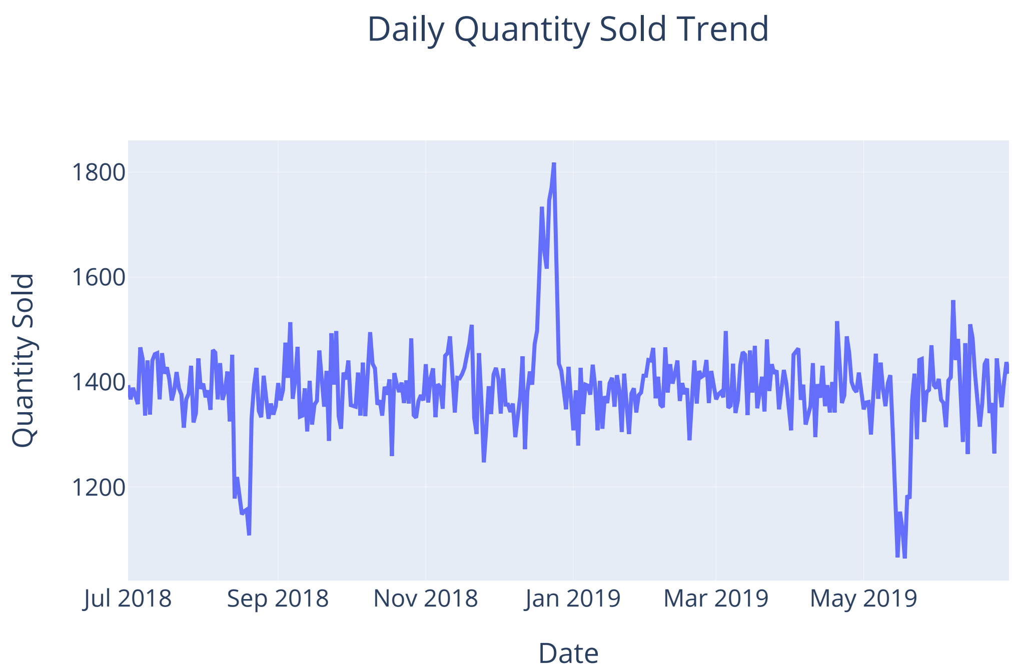
	MONTH-YEAR	PROD_QTY	TOT_SALES	UNIT PRICE
0	2018-07-01	43242	165275.30	86115.60
1	2018-08-01	41284	158081.05	85720.79
2	2018-09-01	41792	160522.00	83385.90
3	2018-10-01	42821	164415.70	85452.30
4	2018-11-01	41895	160233.70	83447.40
5	2018-12-01	43845	167913.40	87294.70
6	2019-01-01	42501	162642.30	84662.10
7	2019-02-01	39220	150665.00	78242.30
8	2019-03-01	43347	166265.20	86539.80
9	2019-04-01	41825	159845.10	83091.30
10	2019-05-01	41100	156717.65	85303.24
11	2019-06-01	41852	160538.60	83636.30

In [56]:

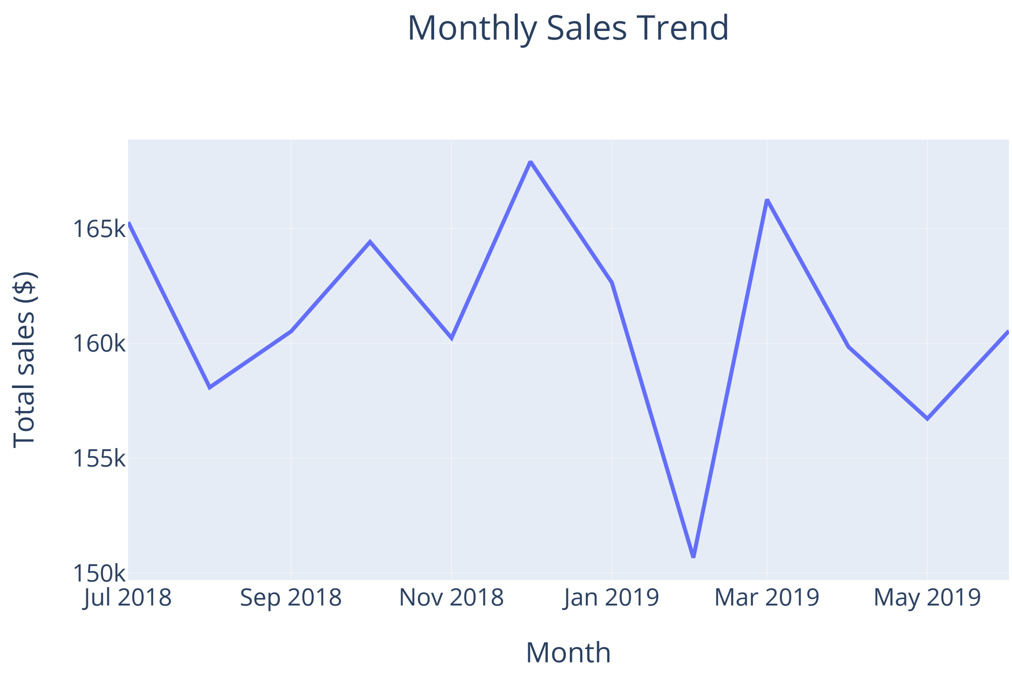
```
#Visualze Daily Sales Trend
fig=px.line(date_trend, x='DATE',y="TOT_SALES",
            title='Daily Sales Trend ',
            labels={'DATE':'Date', "TOT_SALES":"Total Sales ($)"}
           )
fig.update_layout(xaxis_title="Date" ,yaxis_title="Total Sales ($)", width=600,
fig.show()
```



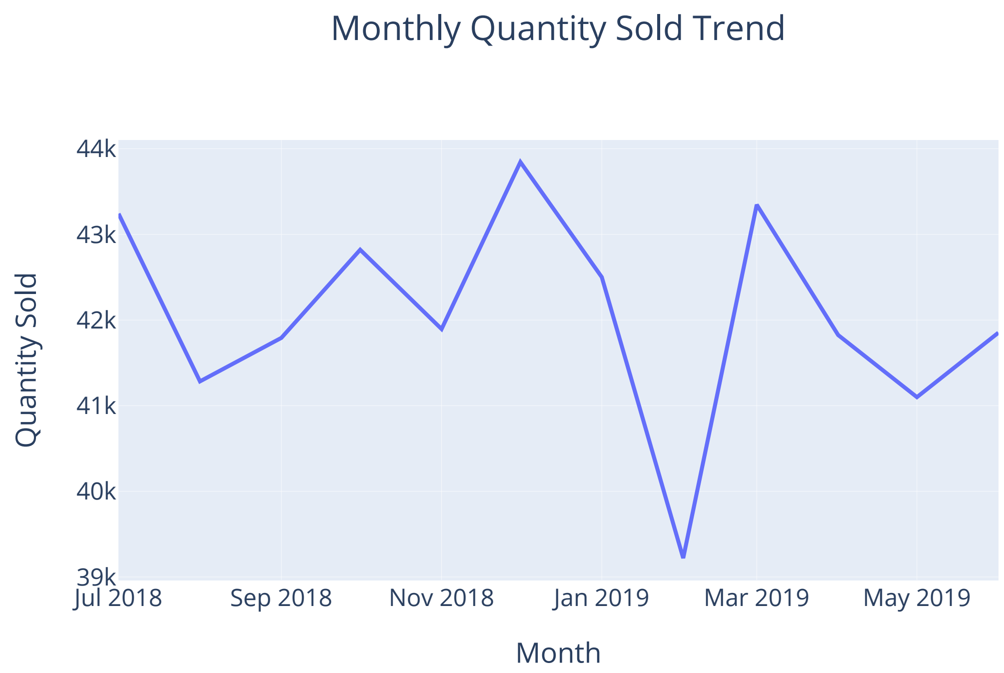
In [57]:
```
#Visualze Daily Product Quantity Trend
fig=px.line(date_trend, x='DATE',y="PROD_QTY",
            title='Daily Quantity Sold Trend ',
            labels={'DATE':'Date', "PROD_QTY":"Quantity Sold"}
           )
fig.update_layout(xaxis_title="Date" ,yaxis_title="Quantity Sold", width=600, he
fig.show()
```



In [58]:
```
#Visualze monthly Total sales Trend
fig=px.line(month_trend, x='MONTH-YEAR',y="TOT_SALES",
            title='Monthly Sales Trend',
            labels={'MONTH-YEAR':'Month',"TOT_SALES":'Total sales ($)'},
           )
fig.update_layout(xaxis_title="Month" ,yaxis_title='Total sales ($)',
                  width=600, height=400,title_x=0.5)
fig.show()
```



In [59]:
```
#Visualze monthly Product Quantity Trend
fig=px.line(month_trend, x='MONTH-YEAR',y="PROD_QTY",
            title='Monthly Quantity Sold Trend',
            labels={'MONTH-YEAR':'Month', "PROD_QTY":"Quantity Sold"}
           )
fig.update_layout(xaxis_title="Month" ,yaxis_title="Quantity Sold",
                  width=600, height=400,title_x=0.5)
fig.show()
```



Key Insights

1. Key Mearsures

- The data was captured from July 2018 to June 2019
- The table below displays the key measures calculated form the data

Measure	Value
Total Sales	$1933115.00
Total Quantity sold	504724 units
Number of Customers	72636
Total Number of Transactions	264836

2. Life Stage by Total Sales

The lifestage column was ranked by the the total sales ($) and the results are displayed in the table below

RANK	LIFESTAGE	TOTAL SALES
1	OLDER SINGLES/COUPLES	402426.75
2	RETIREES	366470.90
3	OLDER FAMILIES	352467.20

RANK	LIFESTAGE	TOTAL SALES
4	YOUNG FAMILIES	316160.10
5	YOUNG SINGLES/COUPLES	260405.30
6	MIDAGE SINGLES/COUPLES	184751.30
7	NEW FAMILIES	50433.45

3. Most Performing Stores

The top 5 Stores by the total sales ($) made by each store is presented in the table below

RANK	STORE NO.	TOTAL SALES
1	226	17605.45
2	88	16333.25
3	165	15973.75
4	40	15559.50
5	237	15539.50

The top 5 Stores by the quantity of products sold by each store is presented in the table below

RANK	STORE NO.	PRODUCT QTY
1	226	4001
2	88	3718
3	93	3639
4	165	3602
5	43	3519

The reason why some stores perform more than others is limited by the data. However, the reasons below may be true

- Top-performing Stores may be located in highly populated areas
- Promotions to improve their sales
- Constant supply of products

4. Top Selling Products

The table below presents the top 5 selling products by total sales ($)

RANK	PRODUCT	TOTAL SALES
1	Dorito Corn Chp Supreme 380g	39052.0

RANK	PRODUCT	TOTAL SALES
2	Smiths Crnkle Chip Orgnl Big Bag 380g	36367.6
3	Smiths Crinkle Chips Salt & Vinegar 330g	34804.2
4	Kettle Mozzarella Basil & Pesto 175g	34457.4
5	Smiths Crinkle Original 330g	34302.6

The table below presents the top 5 selling products by total sales ($)

RANK	PRODUCT	QUANTITY SOLD
1	Kettle Mozzarella Basil & Pesto 175g	6381
2	Kettle Tortilla ChpsHny&Jlpno Chili 150g	6309
3	Cobs Popd Sea Salt Chips 110g	6277
4	Cobs Popd Swt/Chlli &Sr/Cream Chips 110g	6256
5	Tostitos Splash Of Lime 175g	6234

The table below displays the 5 most expensive products.

RANK	PRODUCT	UNIT PRICES ($)
1	Dorito Corn Chp Supreme 380g	6.37
2	Smiths Crnkle Chip Orgnl Big Bag 380g	5.90
3	Doritos Cheese Supreme 330g	5.70
4	Smiths Crinkle Original 330g	5.70
5	Smiths Crinkle Chips Salt & Vinegar 330g	5.70

The main reason why the ranks of the products show the trend above could not be revealed due to data insufficiency. However, the reasons below may be possible.

- The sizes, ingredients, and nutritional value may account for the prices of the products.
- Promotions on certain products and customer preference for them may be the reason why they sell more than others

5. Correlation Analyses

Comparison of the rankings of Store and Products by the **variables**;*Total Sales* and *Quantity sold* revealed that, stores and products ranked differently under the two variables, i.e., stores/products that ranked as the top 5 under *Total sales* are different from the stores/products that rank as the top 5 under *Product Quantity*. A correlation analysis investigated this finding and the result is presented in the table below

CORRELATIONS	PRODUCT QTY	TOTAL SALES	UNIT PRICE
PRODUCT QTY	1.0000	0.5278	0.0271
TOTAL SALES	0.5278	1.0000	0.8498
UNIT PRICE	0.0271	0.8498	1.0000

The correlation analysis results above show that *total sales* correlate more positively with *unit price* than *Product Quantity*. This means that the total sales depend more on the unit prices of the products sold than the quantity of those products sold. This finding also explains why the rankings of stores and products under the variables above differ.

6. Top 5 Customers by Sales

The customers of the company were ranked by the total purchase they made. The result is displayed below

RANK	LOYALTY CARD NO.	TOTAL PURCHASES (Dollars)
1	230078	138.6
2	63197	132.8
3	259009	127.2
4	162039	126.8
5	58361	124.8

7. Customer Financial Satus

The table below shows the distribution of the **72636** customers under customer financial status.

FINANCIAL STATUS	No. of Customers
Mainstream	29245
Budget	24470
Premium	18922

8. Trend Analyses

Monthly Trend

- The monthly sales of the company ranged from **150,665.00 - 167,913.40 dollars**. The company recorded the highest sales in December 2018 and the least sales in February 2019.
- The monthly quantity of products sold the company sold ranged from **39,220 units** to **43,845 units**. Just like sales, the company sold the largest the least quantities in

December 2018 and February 2019 respectively.
- The festivities in December 2018,which influence people to increase their spendinds, is a suspected have caused the highest monthly sales recorded in the month. It is also possible that the company ran and End-of-year promotion which increased sales.
- Possible cause of the lowest monthly sales recorded in February, 2019 could not be assessed

Daily Trend

- Both sales and quantity of products sold follow a steady up-and-down trend. Normally, the company recorded **4,718.50 - 5,924.10 dollars** sales from **1,247 - 1,556 units** of products daily
- Sales and quantity of products sold boomed extremely from 18th - 23rd December, 2018, where the company made more than **6,200 dollars** daily sales from more than **1,600 units** of products
- Just as the market boom, there was and extreme dip from 16th-20th August, 2018 and 14th-20th May,2019 where the company made less than **4,500 dollars** daily sales from less than **1,180 units** of products.
- The daily sales boom recorded on the days stated above is suspected to be linked to the Christmas Festival. Also, the peak daily sales on 23rd December may be due to the fact that, the day preceeded the next two days which were the Chridtmas holidays.
- The possible causes of sales dips on the days stated above are not certain, but may be due to bad internet connection of unfavorable weather conditions such as snow storm or rain.

9. Conclusion an Recomendations

Conclusion

The insights above are true as long as the data is accurate. Analyses to reveal the exact causes behind the findings above con not be done due to paucity of the data.

Recommendations

- Top performing products should be made available at all times.
- It should be ensured that, top performing stores be stocked with enough products. Managers and staff of those stores should be compensated to motivate other stores to improve their performance.
- Leading customers in terms of purchase could be awarded to entice other customers to increase their purchase.
- Further analyses should be done to investigate the actual causes of the insights above.

In [ ]: