```
import pandas as pd
import numpy as np
```

```
df =pd.read_excel('/content/QVI_transaction_data.xlsx')
df2 = pd.read_csv('/content/QVI_purchase_behaviour.csv')
```

```
df.shape
```

```
(264836, 8)
```

```
df2.shape
```

```
(72637, 3)
```

```
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  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
```

```
df['DATE']
```

	DATE
0	43390
1	43599
2	43605
3	43329
4	43330
...	...
264831	43533
264832	43325
264833	43410
264834	43461
264835	43365

264836 rows × 1 columns

dtype: int64

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

```
df['DATE']
```

	DATE
0	2018-10-17
1	2019-05-14
2	2019-05-20
3	2018-08-17
4	2018-08-18
...	...
264831	2019-03-09
264832	2018-08-13
264833	2018-11-06
264834	2018-12-27
264835	2018-09-22

264836 rows × 1 columns

dtype: datetime64[ns]

```
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
```

```
df.isnull().sum()
```

	0
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

```
df2.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
```

```
df2.isnull().sum()
```

	0
LYLTY_CARD_NBR	0
LIFESTAGE	0
PREMIUM_CUSTOMER	0

dtype: int64

```
df.nunique()
```

	0
DATE	364
STORE_NBR	272
LYLTY_CARD_NBR	72637
TXN_ID	263127
PROD_NBR	114
PROD_NAME	114
PROD_QTY	6
TOT_SALES	112

dtype: int64

```
df2.nunique()
```

	0
LYLTY_CARD_NBR	72637
LIFESTAGE	7
PREMIUM_CUSTOMER	3

dtype: int64

```
df.duplicated().sum()
```

```
np.int64(1)
```

```
df[df.duplicated()]
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
124845	2018-10-01	107	107024	108462	45	Smiths Thinly Cut Roast Chicken 175g	2	6.0

```
df2.duplicated().sum()
```

```
np.int64(0)
```

```
df.drop_duplicates(inplace = True)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
df.describe(include = 'all')
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
count	264835	264835.000000	2.648350e+05	2.648350e+05	264835.000000	264835	264835.000000	264835.000000
unique	NaN	NaN	NaN	NaN	NaN	114	NaN	NaN
top	NaN	NaN	NaN	NaN	NaN	Kettle Mozzarella Basil & Pesto 175g	NaN	NaN
freq	NaN	NaN	NaN	NaN	NaN	3304	NaN	NaN
mean	2018-12-30 00:52:42.252722944	135.080216	1.355496e+05	1.351584e+05	56.583201	NaN	1.907308	7.304205
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.000000	NaN	1.000000	1.500000
25%	2018-09-30 00:00:00	70.000000	7.002100e+04	6.760100e+04	28.000000	NaN	2.000000	5.400000
50%	2018-12-30 00:00:00	130.000000	1.303580e+05	1.351380e+05	56.000000	NaN	2.000000	7.400000
75%	2019-03-31 00:00:00	203.000000	2.030945e+05	2.027015e+05	85.000000	NaN	2.000000	9.200000

```
df2.describe(include= 'all')
```

	LYLTY_CARD_NBR	LIFESTAGE	PREMIUM_CUSTOMER
count	7.263700e+04	72637	72637
unique	NaN	7	3
top	NaN	RETIREES	Mainstream
freq	NaN	14805	29245
mean	1.361859e+05	NaN	NaN
std	8.989293e+04	NaN	NaN
min	1.000000e+03	NaN	NaN
25%	6.620200e+04	NaN	NaN
50%	1.340400e+05	NaN	NaN
75%	2.033750e+05	NaN	NaN
max	2.373711e+06	NaN	NaN

```
Q1 = df['TOT_SALES'].quantile(0.25)
Q3 = df['TOT_SALES'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers_sales = df[(df['TOT_SALES'] < lower_bound) | (df['TOT_SALES'] > upper_bound)]

print("Outliers in TOT_SALES:", outliers_sales.shape[0])
```

```
Outliers in TOT_SALES: 576
```

```
outliers_sales
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
3	2018-08-17	2	2373	974	69	Smiths Chip Thinly S/Cream&Onion 175g	5	15.0
11	2018-08-20	8	8294	8221	114	Kettle Sensations Siracha Lime 150g	5	23.0
56	2019-05-16	74	74336	73182	84	GrnWves Plus Btroot & Chilli Jam 180g	5	15.5
72	2018-08-19	96	96203	96025	7	Smiths Crinkle Original 330g	5	28.5
100	2019-05-20	130	130108	134125	2	Cobs Popd Sour Crm &Chives Chips 110g	5	19.0
...	...	...	...	...	...	...	...	...
258715	2018-08-16	194	194381	194835	102	Kettle Mozzarella Basil & Pesto 175g	4	21.6
258721	2018-08-15	200	200248	199694	3	Kettle Sensations Camembert & Fig 150g	4	18.4
258726	2018-08-20	203	203253	203360	28	Thins Potato Chips Hot & Spicy 175g	5	16.5
258729	2019-05-16	208	208205	207318	37	Smiths Thinly Swt Chli&S/Cream175G	5	15.0
258788	2019-05-14	264	264149	262909	25	Pringles SourCream Onion 134g	5	18.5

576 rows × 8 columns

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

```
outliers_sales['TOT_SALES'].value_counts(ascending = False)
```

TOT_SALES	count
18.50	50
23.00	39
15.00	38
18.40	35
27.00	35
19.00	31
22.00	30
21.60	30
17.60	29
16.50	27
22.80	26
16.20	25
17.10	22
15.20	21
25.50	14
20.40	14
28.50	12
19.50	12
15.30	12
16.80	10
21.00	10
17.70	9
18.00	7
16.25	7
29.50	7
15.60	6
17.20	6
21.50	5
23.60	4
15.50	3

dtype: int64

```
df[df['TOT_SALES']==650]
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES

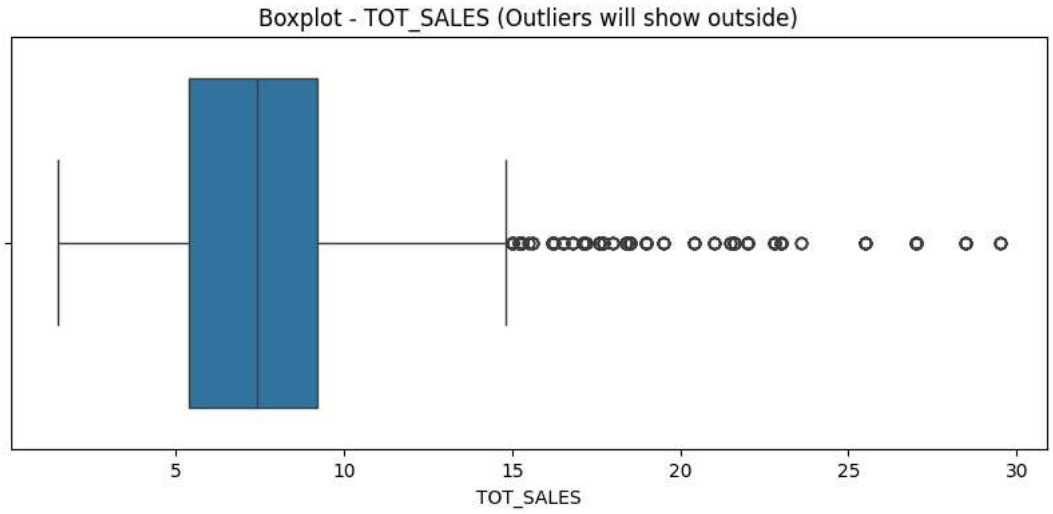
```
df[df['LYLTY_CARD_NBR']==226000]
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES
69762	2018-08-19	226	226000	226201	4	Dorito Corn Chp Supreme 380g	200	650.0
69763	2019-05-20	226	226000	226210	4	Dorito Corn Chp Supreme 380g	200	650.0

```
df.drop(df[df['LYLTY_CARD_NBR']==226000].index, inplace = True)
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(10,4))
sns.boxplot(x=df['TOT_SALES'])
plt.title("Boxplot - TOT_SALES (Outliers will show outside)")
plt.show()
```




```
Q1 = df['PROD_QTY'].quantile(0.25)
Q3 = df['PROD_QTY'].quantile(0.75)
IQR = Q3 - Q1

lower_q = Q1 - 1.5 * IQR
upper_q = Q3 + 1.5 * IQR

outliers_qty = df[(df['PROD_QTY'] < lower_q) | (df['PROD_QTY'] > upper_q)]

print("Outliers in PROD_QTY:", outliers_qty.shape[0])
```

```
Outliers in PROD_QTY: 28795
```

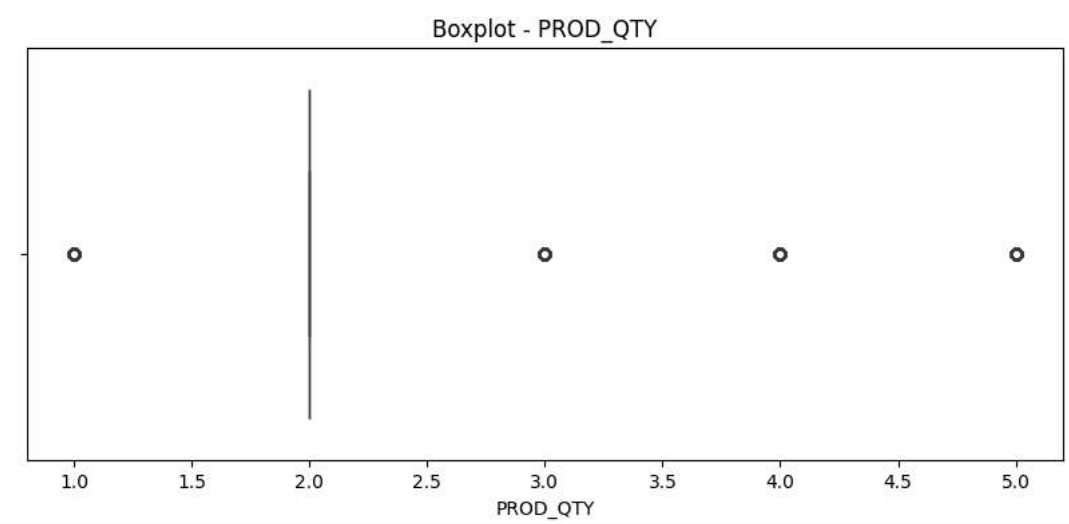
```
Start coding or generate with AI.
```

```
outliers_qty['PROD_QTY'].value_counts(ascending = False)
```

PROD_QTY	count
1	27518
5	450
3	430
4	397

dtype: int64

```
plt.figure(figsize=(10,4))
sns.boxplot(x=df['PROD_QTY'])
plt.title("Boxplot - PROD_QTY")
plt.show()
```




```
def detect_outliers_iqr(data, col):
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    outliers = data[(data[col] < lower) | (data[col] > upper)]
    return outliers

numeric_cols = ['TOT_SALES', 'PROD_QTY']

for col in numeric_cols:
    outliers = detect_outliers_iqr(df, col)
    print(f"{col}: {len(outliers)} outliers")
```

```
TOT_SALES: 576 outliers
PROD_QTY: 28795 outliers
```

```
merged_df = pd.merge(df, df2, on='LYLTY_CARD_NBR', how='inner')
```

```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264833 entries, 0 to 264832
Data columns (total 10 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   DATE              264833 non-null  datetime64[ns]
 1   STORE_NBR         264833 non-null  int64
 2   LYLTY_CARD_NBR    264833 non-null  int64
 3   TXN_ID            264833 non-null  int64
 4   PROD_NBR          264833 non-null  int64
 5   PROD_NAME         264833 non-null  object
 6   PROD_QTY          264833 non-null  int64
 7   TOT_SALES         264833 non-null  float64
 8   LIFESTAGE         264833 non-null  object
 9   PREMIUM_CUSTOMER  264833 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(5), object(3)
memory usage: 20.2+ MB
```

```
merged_df.describe(include = 'all')
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_NAME	PROD_QTY	TOT_SALES	LI
count	264833	264833.000000	2.648330e+05	2.648330e+05	264833.000000	264833	264833.000000	264833.000000	
unique	NaN	NaN	NaN	NaN	NaN	114	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	Kettle Mozzarella Basil & Pesto 175g	NaN	NaN	SINGLES/C
freq	NaN	NaN	NaN	NaN	NaN	3304	NaN	NaN	
mean	2018-12-30 00:52:39.666657792	135.079529	1.355489e+05	1.351577e+05	56.583598	NaN	1.905812	7.299351	
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.000000	NaN	1.000000	1.500000	
25%	2018-09-30 00:00:00	70.000000	7.002100e+04	6.760000e+04	28.000000	NaN	2.000000	5.400000	
50%	2018-12-30 00:00:00	130.000000	1.303570e+05	1.351370e+05	56.000000	NaN	2.000000	7.400000	

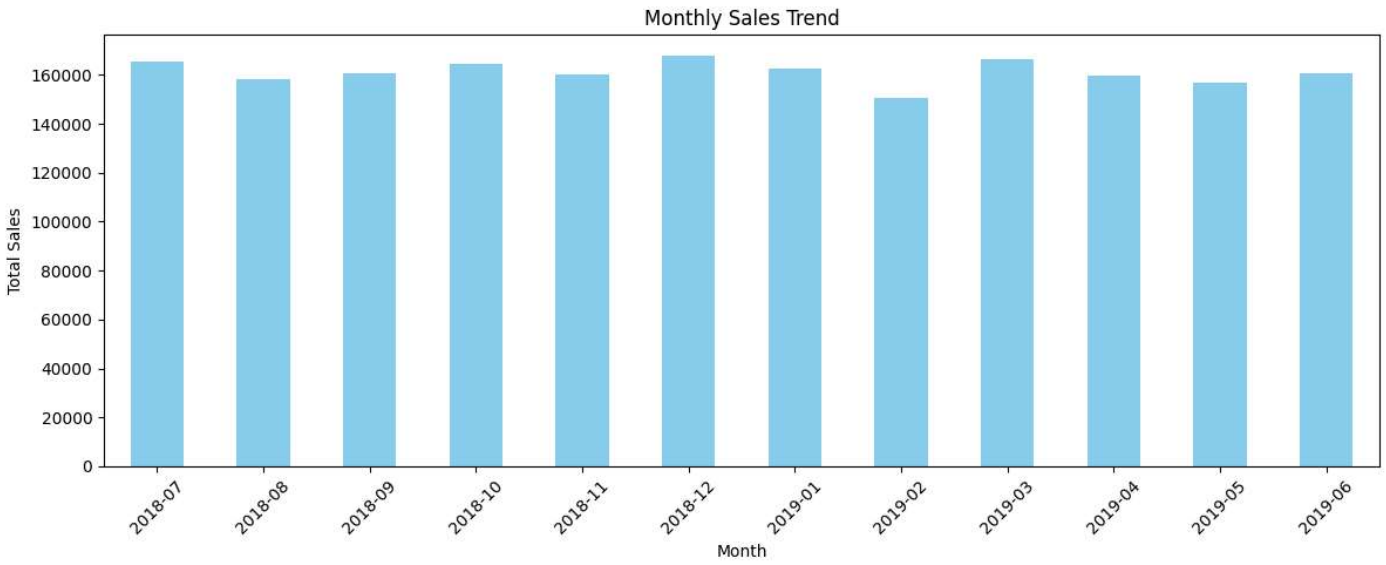
```
for col in ['TOT_SALES', 'PROD_QTY']:
    Q1 =merged_df[col].quantile(0.25)
    Q3 = merged_df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    mild_outliers = merged_df[(merged_df[col] < lower) | (merged_df[col] > upper)]
    print(f"{col}: {len(mild_outliers)} mild outliers")
```

```
TOT_SALES: 576 mild outliers
PROD_QTY: 28795 mild outliers
```

monthly sales trend

```
merged_df['MONTH'] = merged_df['DATE'].dt.to_period('M')
monthly_sales = merged_df.groupby('MONTH')['TOT_SALES'].sum()

monthly_sales.plot(kind='bar', figsize=(12,5), color='skyblue', title='Monthly Sales Trend')
plt.xlabel("Month")
plt.ylabel("Total Sales")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



1. Monthly Sales Trend Observation:

-Sales remained fairly consistent between July 2018 to June 2019.

-Peaks observed in Dec 2018 and Mar 2019, indicating seasonal effects (likely holidays or promotions).

-Slight dip in Feb 2019, possibly due to fewer days in the month or post-holiday slowdown.

- Insights:

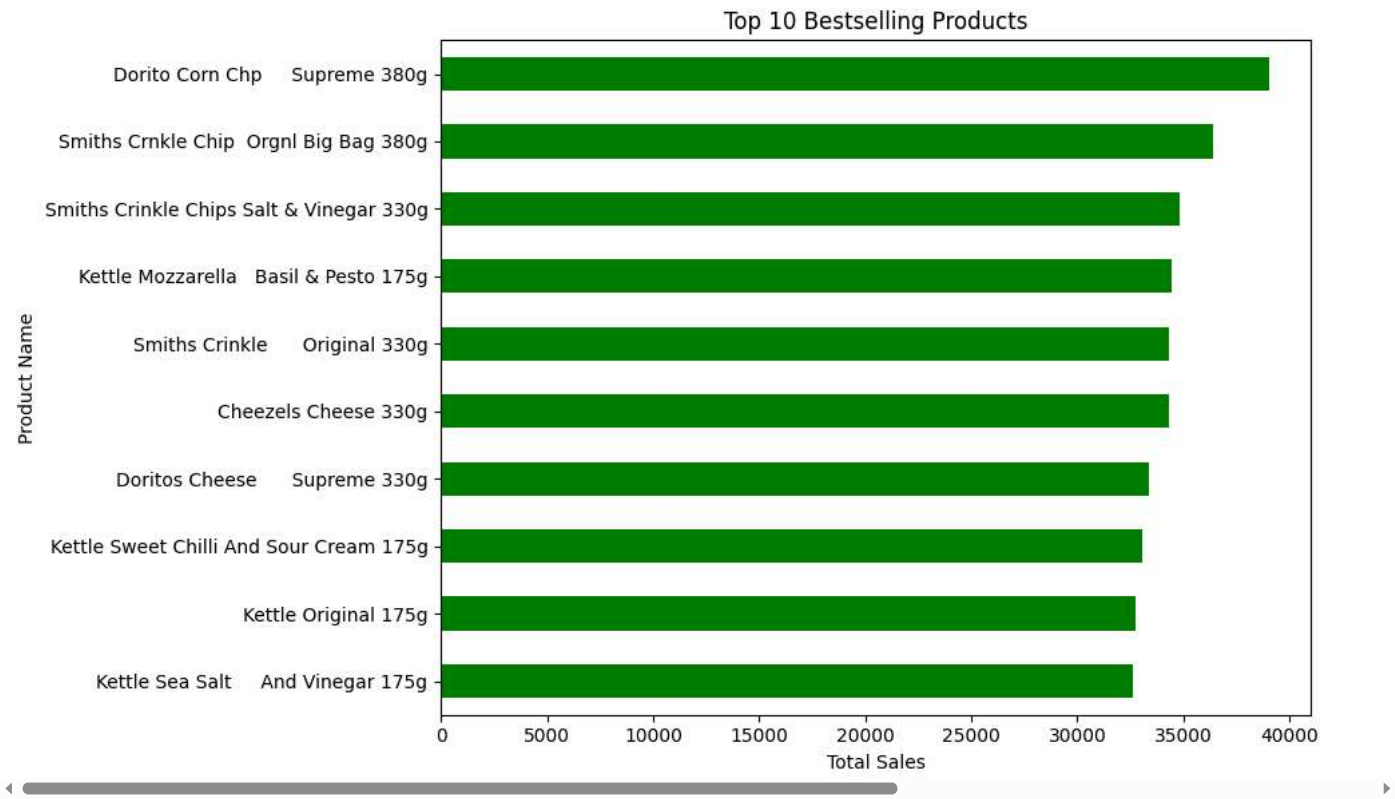
-December and March can be targeted for promotional campaigns.

-Investigate why February had lower sales – was it fewer operational days, stock issues, or reduced footfall?

Top 10 bestselling product

```
top_products = merged_df.groupby('PROD_NAME')['TOT_SALES'].sum().sort_values(ascending=False).head(10)

top_products.plot(kind='barh', figsize=(10,6), color='green', title='Top 10 Bestselling Products')
plt.xlabel("Total Sales")
plt.ylabel("Product Name")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



2. Top 10 Bestselling Products Observation:

-Dorito Corn Chp Supreme 380g is the top-selling product.

-All top 10 products are variants of chips/snacks.

-Products are largely dominated by brands like Kettle, Smiths, Doritos.

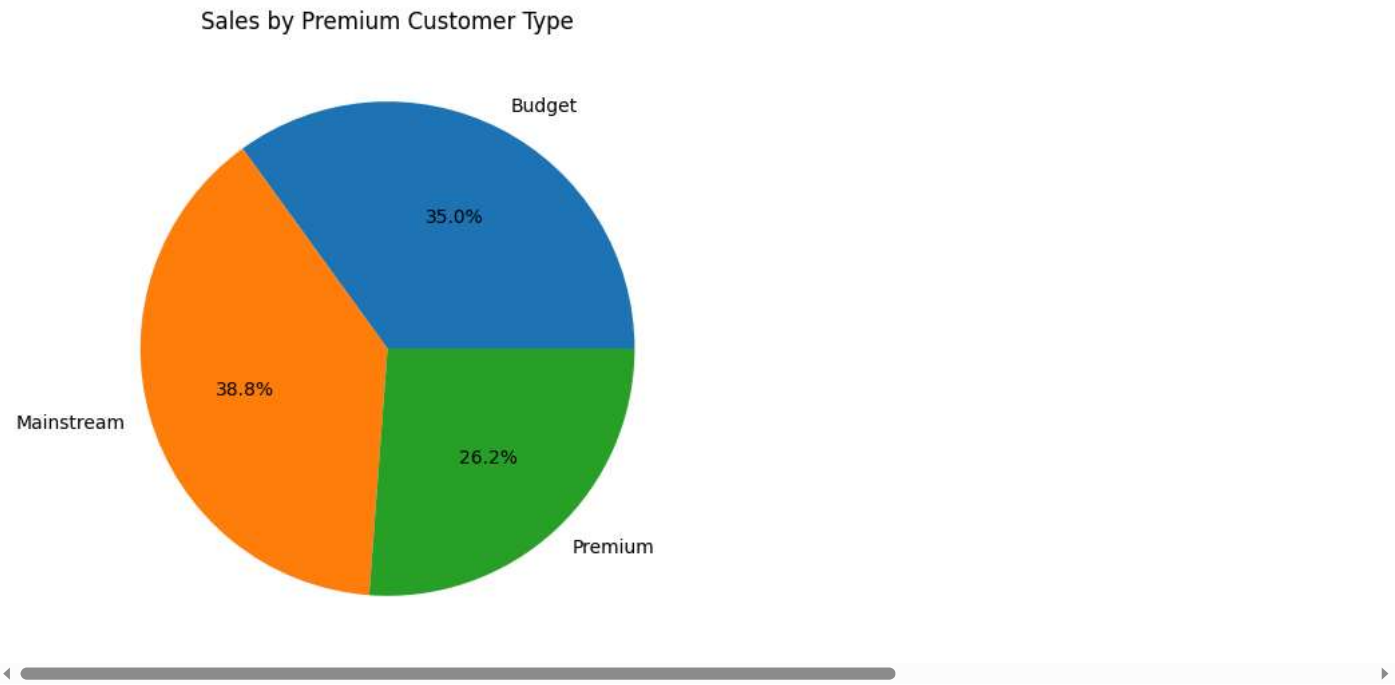
- Insights:

-These products are your cash cows – focus on inventory availability, cross-selling, and upselling during checkout.

-Bundle top-selling products with slower-moving ones to balance stock.

Total Sales over Premium customer

```
merged_df.groupby('PREMIUM_CUSTOMER')['TOT_SALES'].sum().plot(kind='pie', autopct='%1.1f%%', figsize=(6,6), title='Sales by Premium Customer
plt.ylabel('')
plt.show()
```



- Sales by Premium Customer Type Observation:

-Mainstream customers contribute the most to sales (~39%).

-Budget customers (~35%) also form a major chunk.

-Premium customers contribute least (~26%).

- Insights:

-While premium customers spend more per transaction, their frequency might be low.

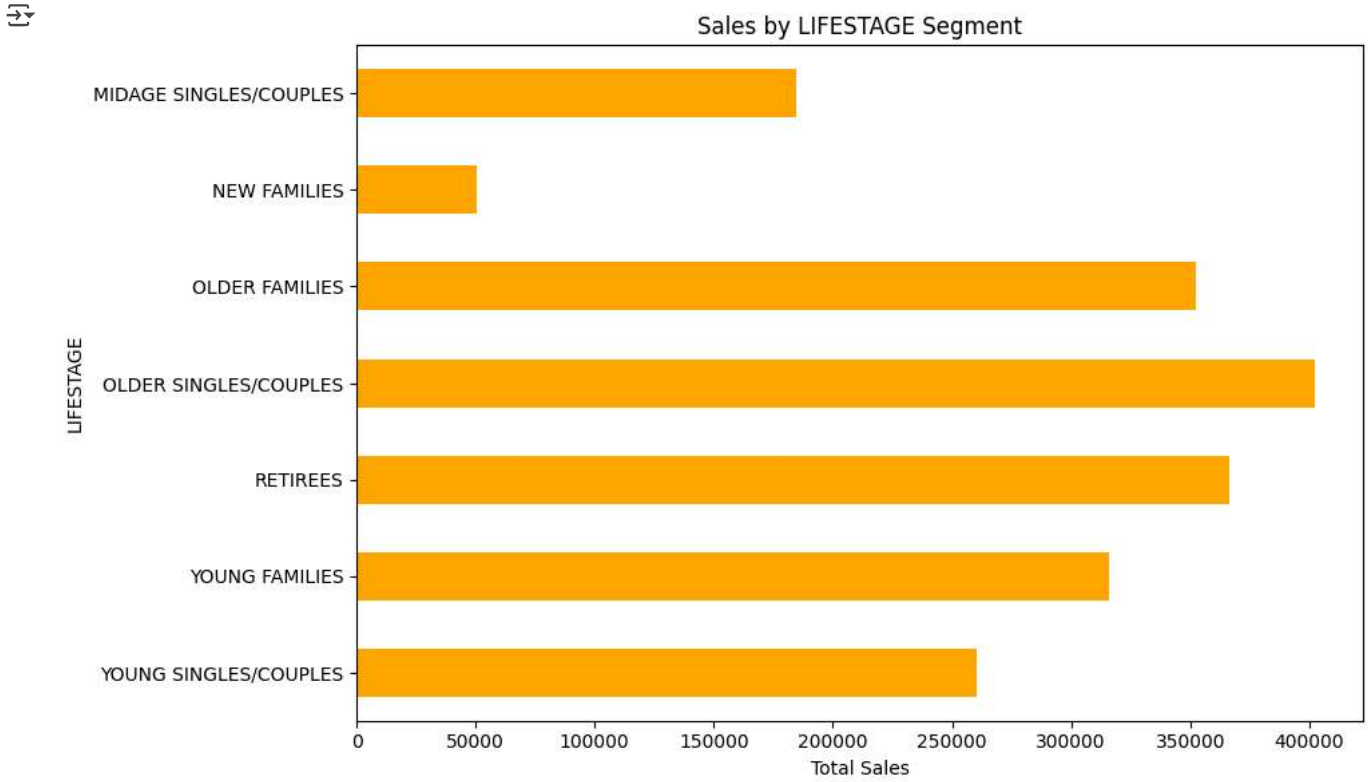
-You can design loyalty programs or exclusive offers to retain premium customers and encourage budget customers to upgrade.

Sales over LifeStage

```
merged_df.groupby('LIFESTAGE')['TOT_SALES'].sum().plot(kind='barh', figsize=(10,6), color='orange', title='Sales by LIFESTAGE Segment')
plt.xlabel("Total Sales")
plt.ylabel("LIFESTAGE")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```

Sales over LifeStage

```
merged_df.groupby('LIFESTAGE')['TOT_SALES'].sum().plot(kind='barh', figsize=(10,6), color='orange', title='Sales by LIFESTAGE Segment')
plt.xlabel("Total Sales")
plt.ylabel("LIFESTAGE")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



4. Sales by LIFESTAGE Segment

- Observation:

-OLDER SINGLES/COUPLES and RETIREES contribute the most.

-NEW FAMILIES and MIDAGE SINGLES/COUPLES contribute the least.

- Insights:

-Older demographic is more engaged — possibly due to brand loyalty or routine-based consumption.

-Target campaigns toward young families and mid-age singles with combo offers or family-sized packs.

avg spend per transaction

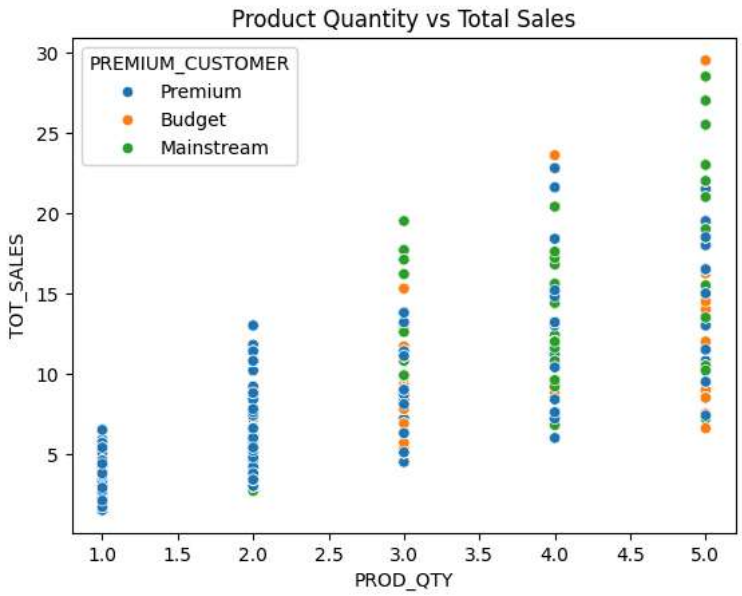
```
avg_spend = merged_df.groupby('LYLTY_CARD_NBR')['TOT_SALES'].sum() / merged_df.groupby('LYLTY_CARD_NBR')['TXN_ID'].nunique()
avg_spend.describe()
```

	0
count	72636.000000
mean	7.133731
std	2.167619
min	1.500000
25%	5.900000
50%	7.400000
75%	8.600000
max	29.500000

dtype: float64

quantity vs Sales Distribution

```
sns.scatterplot(data=merged_df, x='PROD_QTY', y='TOT_SALES', hue='PREMIUM_CUSTOMER')
plt.title("Product Quantity vs Total Sales")
plt.show()
```



5. Product Quantity vs Total Sales by Customer Type
    - Observation:

-Most purchases cluster between 1 to 5 quantity.

-There is a positive trend: more quantity = more total sales, across all segments.

-No significant difference in purchase quantity among premium, budget, or mainstream customers.

- Insights:

-Try volume-based discounts (Buy 3 Get 1 Free) to push customers toward larger quantities.

-Analyze per customer spending behavior for personalized recommendations.

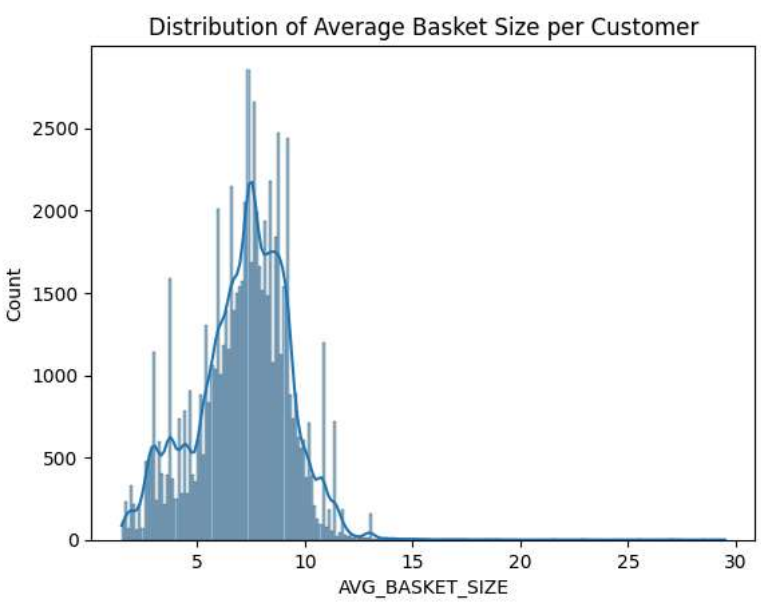
customer segmentation

```
df_cust = merged_df.groupby('LYLTY_CARD_NBR').agg({
    'TOT_SALES':'sum',
    'TXN_ID':'nunique',
    'PROD_QTY':'sum'
```

```
}).rename(columns={'TXN_ID':'NUM_TXNS'})

df_cust['AVG_BASKET_SIZE'] = df_cust['TOT_SALES'] / df_cust['NUM_TXNS']

# Visualize
sns.histplot(df_cust['AVG_BASKET_SIZE'], kde=True)
plt.title("Distribution of Average Basket Size per Customer")
plt.show()
```



6. Average Basket Size Distribution

- Observation:

-The majority of customers have a basket size between 5 to 10 units.

-Long tail observed – very few customers buy more than 15 units.

- Insights:

-Majority customers shop for daily or weekly consumption.

-Create "basket size offers" – e.g., spend ₹500 and get ₹50 off – to move customers to the next tier.

Start coding or generate with AI.