

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
```

```
transaction_df = pd.read_excel("QVI_transaction_data.xlsx")
customer_df = pd.read_csv("QVI_purchase_behaviour.csv")
```

```
transaction_df.head()
transaction_df.info()
transaction_df.describe()
```

<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

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES
count	264836.000000	264836.00000	2.648360e+05	2.648360e+05	264836.000000	264836.000000	264836.000000
mean	43464.036260	135.08011	1.355495e+05	1.351583e+05	56.583157	1.907309	7.304200
std	105.389282	76.78418	8.057998e+04	7.813303e+04	32.826638	0.643654	3.083226
min	43282.000000	1.00000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.500000
25%	43373.000000	70.00000	7.002100e+04	6.760150e+04	28.000000	2.000000	5.400000
50%	43464.000000	130.00000	1.303575e+05	1.351375e+05	56.000000	2.000000	7.400000
75%	43555.000000	203.00000	2.030942e+05	2.027012e+05	85.000000	2.000000	9.200000
max	43646.000000	272.00000	2.373711e+06	2.415841e+06	114.000000	200.000000	650.000000

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

```
transaction_df["PROD_NAME"].nunique()
transaction_df["PROD_NAME"].head()
```

PROD_NAME

0	Natural Chip Compny SeaSalt175g
1	CCs Nacho Cheese 175g
2	Smiths Crinkle Cut Chips Chicken 170g
3	Smiths Chip Thinly S/Cream&Onion 175g
4	Kettle Tortilla ChpsHny&Jlpno Chili 150g

dtype: object

```
transaction_df = transaction_df[
    ~transaction_df["PROD_NAME"].str.lower().str.contains("salsa")
]
transaction_df.isnull().sum()
transaction_df.describe()
```

	DATE	STORE_NBR	LYLTY_CARD_NBR	TXN_ID	PROD_NBR	PROD_QTY	TOT_SALES
count	246742	246742.000000	2.467420e+05	2.467420e+05	246742.000000	246742.000000	246742.000000
mean	2018-12-30 01:19:01.211467520	135.051098	1.355310e+05	1.351311e+05	56.351789	1.908062	7.321322
min	2018-07-01 00:00:00	1.000000	1.000000e+03	1.000000e+00	1.000000	1.000000	1.700000
25%	2018-09-30 00:00:00	70.000000	7.001500e+04	6.756925e+04	26.000000	2.000000	5.800000
50%	2018-12-30 00:00:00	130.000000	1.303670e+05	1.351830e+05	53.000000	2.000000	7.400000
75%	2019-03-31 00:00:00	203.000000	2.030840e+05	2.026538e+05	87.000000	2.000000	8.800000
max	2019-06-30 00:00:00	272.000000	2.373711e+06	2.415841e+06	114.000000	200.000000	650.000000

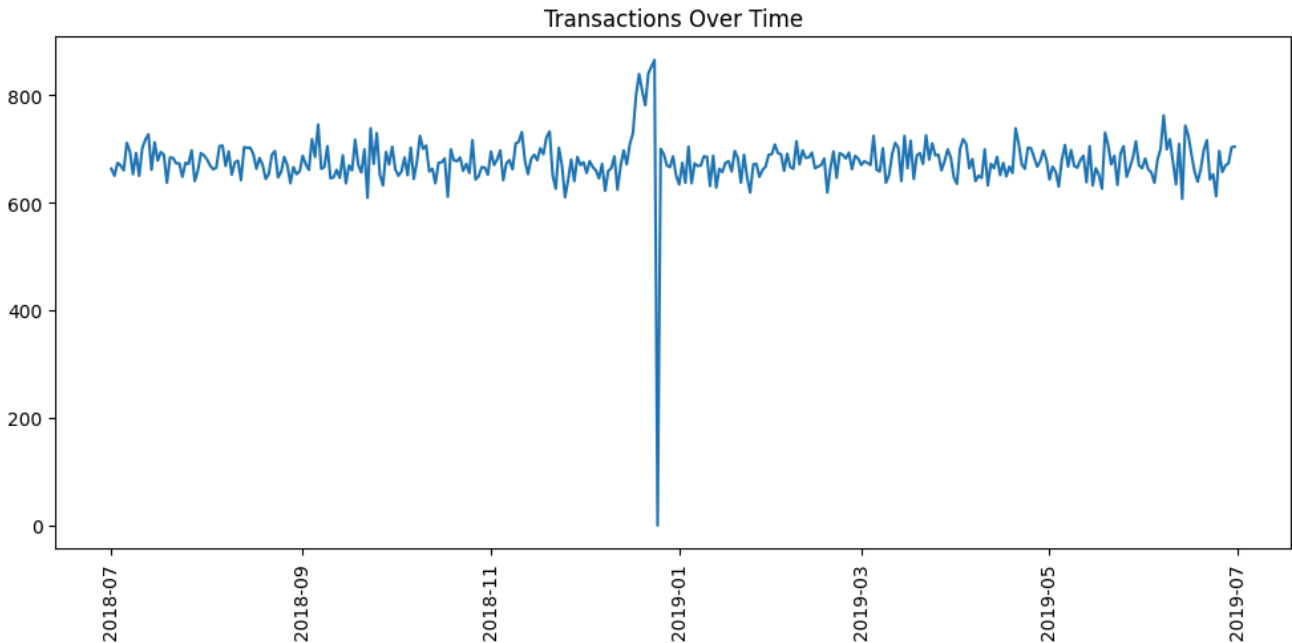
```
transaction_df[transaction_df["PROD_QTY"] == 200]
```

	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

```
transaction_df = transaction_df[
    transaction_df["LYLTY_CARD_NBR"] != 226000
]
transactions_by_day = transaction_df.groupby("DATE").size().reset_index(name="COUNT")
all_dates = pd.date_range("2018-07-01", "2019-06-30")

transactions_by_day = (
    pd.DataFrame({"DATE": all_dates})
    .merge(transactions_by_day, on="DATE", how="left")
    .fillna(0)
)
```

```
plt.figure(figsize=(12,5))
plt.plot(transactions_by_day["DATE"], transactions_by_day["COUNT"])
plt.xticks(rotation=90)
plt.title("Transactions Over Time")
plt.show()
```

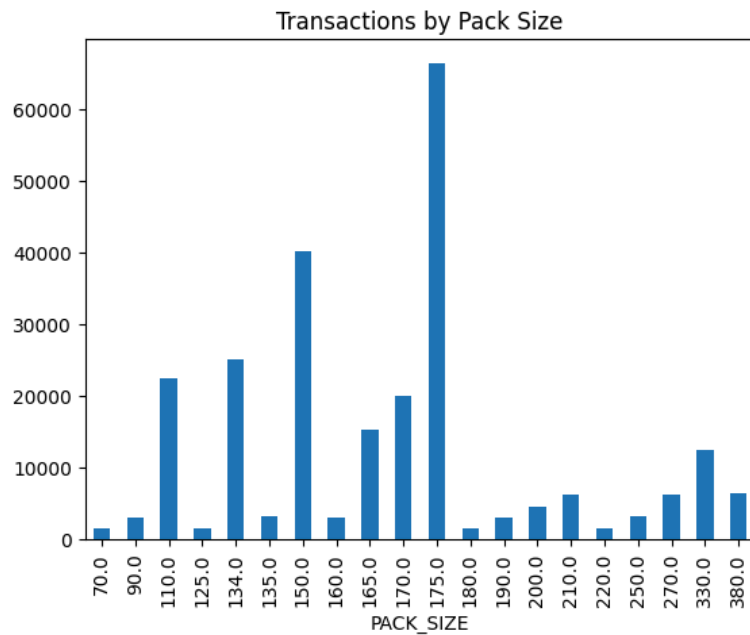


```
transaction_df["PACK_SIZE"] = transaction_df["PROD_NAME"].str.extract(r'(\d+)').astype(float)
transaction_df["BRAND"] = transaction_df["PROD_NAME"].str.split().str[0].str.upper()
transaction_df["BRAND"] = transaction_df["BRAND"].replace({"
```

```

"RED": "RRD",
"SNBTS": "SUNBITES",
"WW": "WOOLWORTHS"
})
transaction_df["PACK_SIZE"].value_counts().sort_index().plot(kind="bar")
plt.title("Transactions by Pack Size")
plt.show()

```



```

customer_df.info()
customer_df["LIFESTAGE"].value_counts()
customer_df["PREMIUM_CUSTOMER"].value_counts()

```

```

<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

```

```

count
PREMIUM_CUSTOMER
Mainstream    29245
Budget        24470
Premium       18922

```

```
dtype: int64
```

```

data = transaction_df.merge(customer_df, on="LYLTY_CARD_NBR", how="left")
data.isnull().sum()

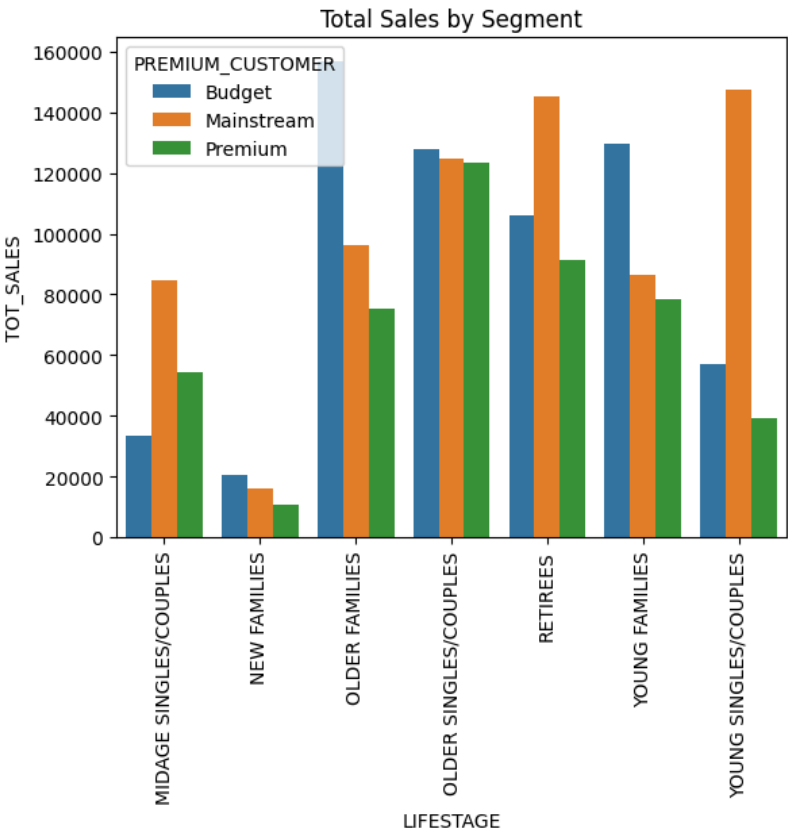
```

	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
PACK_SIZE	0
BRAND	0
LIFESTAGE	0
PREMIUM_CUSTOMER	0

dtype: int64

```
data["PRICE_PER_UNIT"] = data["TOT_SALES"] / data["PROD_QTY"]
sales_segment = data.groupby(
    ["LIFESTAGE", "PREMIUM_CUSTOMER"]
)["TOT_SALES"].sum().reset_index()
sns.barplot(data=sales_segment,
            x="LIFESTAGE",
            y="TOT_SALES",
            hue="PREMIUM_CUSTOMER")

plt.xticks(rotation=90)
plt.title("Total Sales by Segment")
plt.show()
```



```
customer_segment = data.groupby(
    ["LIFESTAGE", "PREMIUM_CUSTOMER"]
)["LYLTY_CARD_NBR"].nunique().reset_index()
```

```
from scipy.stats import ttest_ind

mainstream = data[data["PREMIUM_CUSTOMER"] == "Mainstream"]["PRICE_PER_UNIT"]
others = data[data["PREMIUM_CUSTOMER"] != "Mainstream"]["PRICE_PER_UNIT"]

ttest_ind(mainstream, others)

TtestResult(statistic=np.float64(15.051481410694292), pvalue=np.float64(3.557604503522995e-51), df=np.float64(246738.0))
```

```
target_segment = data[
    (data["LIFESTAGE"] == "YOUNG SINGLES/COUPLES") &
    (data["PREMIUM_CUSTOMER"] == "Mainstream")
]
target_segment["BRAND"].value_counts()
```

count	
BRAND	
KETTLE	3844
PRINGLES	2315
DORITOS	2076
SMITHS	1790
THINS	1166
INFUZIONI	962
TWISTIES	900
TOSTITOS	890
RRD	875
COBS	864
TYRRELLS	619
GRAIN	576
WOOLWORTHS	479
CHEEZELS	346
NATURAL	321
DORITO	303
INFZNS	288
CCS	222
CHEETOS	166
SMITH	131
SUNBITES	128
FRENCH	78
NCC	73
GRNWVES	70
BURGER	62

dtype: int64

```
target_segment["PACK_SIZE"].value_counts()
```

	count
PACK_SIZE	
175.0	4997
150.0	3080
134.0	2315
110.0	2051
170.0	1575
330.0	1195
165.0	1102
380.0	626
270.0	620
210.0	576
135.0	290
250.0	280
200.0	179
190.0	148
90.0	128
160.0	128
180.0	70
70.0	63
220.0	62
125.0	59

dtype: int64

data.to_csv("QVI_clean_data.csv", index=False)

Quantum Customer Behaviour Analysis Insights

1 High Value Customer Segment

The analysis identified **Young Singles/Couples within the Mainstream segment** as one of the highest contributors to chip sales. This group demonstrates frequent purchasing behaviour and represents a large proportion of total transactions.

Statistical testing confirms that their purchasing behaviour differs significantly from other segments (p-value ≈ 0).

Business Implication

This segment represents a high-value customer group and should be prioritised in targeted marketing strategies.

2 Price Sensitivity & Spending Behaviour

Analysis shows that mainstream customers tend to pay a higher price per unit compared to other customer groups. This indicates lower price sensitivity and stronger brand preference.

Business Implication

There is an opportunity to introduce premium chip products targeting this segment to maximise profit margins.

3 Brand Preference Analysis

The most frequently purchased chip brands were:

- Kettle
- Pringles

- Doritos
- Smiths
- Thins

Business Implication

These brands should receive priority shelf placement and promotional visibility.

4 Pack Size Purchasing Behaviour

Most popular pack sizes include:

- 175g (Most popular)
- 150g
- 134g
- 110g

Business Implication

Mid-sized packs balance price and convenience and should be prioritised.

5 Sales Trend Analysis

Transaction data shows stable demand for chip products across the year.

Business Implication

Chips represent a consistent revenue category for retailers.