Chips Category Analysis

Introduction

The objective of this project is to analyze customer purchase behaviour and transaction data for the chips category, identify key drivers of sales, and provide actionable recommendations to the Category Manager.

The analysis follows a structured workflow:

1. **Data Preparation** – cleaning and merging customer and transaction datasets.
2. **Exploratory Analysis** – examining sales by customer segments and packet sizes.
3. **Segment Drivers** – understanding what drives sales (frequency vs spend).
4. **Visualization** – creating clear charts to communicate insights.
5. **Insights & Recommendations** – defining strategies for targeting the right customer segments and packet sizes.

Import & Setup

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings("ignore")

# Configure plots
sns.set_theme(style="whitegrid")   # Seaborn handles styling
sns.set_palette("Set2")            # Nice pastel palette
```

```
# For manual file upload in Colab
from google.colab import files
import pandas as pd

# Upload files (you'll get a "Choose Files" button in Colab)
uploaded = files.upload()
```

Choose Files 2 files
QVI_purchase_behaviour.csv(text/csv) - 2452463 bytes, last modified: 28/11/2025 - 100% done
QVI_transaction_data.xlsx(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 11979155 bytes, last modified: 28/11/2025 - 100% done

```
Saving QVI_purchase_behaviour.csv to QVI_purchase_behaviour.csv
Saving QVI_transaction_data.xlsx to QVI_transaction_data.xlsx
```

Data Loading

```
# Load CSV
purchase_behaviour = pd.read_csv("QVI_purchase_behaviour.csv")

# Load Excel
transaction_data = pd.read_excel("QVI_transaction_data.xlsx")

# Quick inspection
print("Purchase Behaviour Data:")
print(purchase_behaviour.head(), "\n")

print("Transaction Data:")
print(transaction_data.head())
```

```
Purchase Behaviour Data:
   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

Transaction Data:
    DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
```

```
   0  43390          1           1000       1         5
   1  43599          1           1307     348        66
   2  43605          1           1343     383        61
   3  43329          2           2373     974        69
   4  43330          2           2426    1038       108

                                 PROD_NAME  PROD_QTY  TOT_SALES
   0    Natural Chip        Compny SeaSalt175g         2        6.0
   1                  CCs Nacho Cheese    175g         3        6.3
   2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
   3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
   4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8
```

Data Inspection

```python
# Check missing values
print("Purchase Behaviour Missing Values:\n", purchase_behaviour.isnull().sum())
print("\nTransaction Data Missing Values:\n", transaction_data.isnull().sum())

# Check duplicates
print("\nPurchase Behaviour Duplicates:", purchase_behaviour.duplicated().sum())
print("Transaction Data Duplicates:", transaction_data.duplicated().sum())

# Quick stats for numeric columns
print("\nTransaction Data Stats:\n", transaction_data.describe())
```

```
Purchase Behaviour Missing Values:
 LYLTY_CARD_NBR      0
LIFESTAGE           0
PREMIUM_CUSTOMER    0
dtype: int64

Transaction Data Missing Values:
 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

Purchase Behaviour Duplicates: 0
Transaction Data Duplicates: 1

Transaction Data Stats:
                 DATE     STORE_NBR  LYLTY_CARD_NBR        TXN_ID  \
count  264836.000000  264836.00000    2.648360e+05  2.648360e+05
mean    43464.036260     135.08011    1.355495e+05  1.351583e+05
std       105.389282      76.78418    8.057998e+04  7.813303e+04
min     43282.000000       1.00000    1.000000e+03  1.000000e+00
25%     43373.000000      70.00000    7.002100e+04  6.760150e+04
50%     43464.000000     130.00000    1.303575e+05  1.351375e+05
75%     43555.000000     203.00000    2.030942e+05  2.027012e+05
max     43646.000000     272.00000    2.373711e+06  2.415841e+06

            PROD_NBR       PROD_QTY      TOT_SALES
count  264836.000000  264836.000000  264836.000000
mean       56.583157       1.907309       7.304200
std        32.826638       0.643654       3.083226
min         1.000000       1.000000       1.500000
25%        28.000000       2.000000       5.400000
50%        56.000000       2.000000       7.400000
75%        85.000000       2.000000       9.200000
max       114.000000     200.000000     650.000000
```

Data Cleaning

```python
# Drop duplicate row in transaction data
transaction_data.drop_duplicates(inplace=True)
```

```python
# Remove extreme outliers in product quantity and sales
transaction_data = transaction_data[transaction_data['PROD_QTY'] < 50]
transaction_data = transaction_data[transaction_data['TOT_SALES'] < 100]
```

Data Merge

```
merged = pd.merge(transaction_data, purchase_behaviour, on="LYLTY_CARD_NBR", how="inner")

print("Merged Data Sample:\n", merged.head())
```

```
Merged Data Sample:
     DATE  STORE_NBR  LYLTY_CARD_NBR  TXN_ID  PROD_NBR  \
0  43390          1            1000       1         5
1  43599          1            1307     348        66
2  43605          1            1343     383        61
3  43329          2            2373     974        69
4  43330          2            2426    1038       108

                                  PROD_NAME  PROD_QTY  TOT_SALES  \
0    Natural Chip        Compny SeaSalt175g         2        6.0
1                  CCs Nacho Cheese    175g         3        6.3
2    Smiths Crinkle Cut  Chips Chicken 170g         2        2.9
3    Smiths Chip Thinly  S/Cream&Onion 175g         5       15.0
4  Kettle Tortilla ChpsHny&Jlpno Chili 150g         3       13.8

               LIFESTAGE PREMIUM_CUSTOMER
0   YOUNG SINGLES/COUPLES          Premium
1  MIDAGE SINGLES/COUPLES           Budget
2  MIDAGE SINGLES/COUPLES           Budget
3  MIDAGE SINGLES/COUPLES           Budget
4  MIDAGE SINGLES/COUPLES           Budget
```

Core Analysis

```
# Sales by Lifestage
lifestage_sales = merged.groupby("LIFESTAGE")['TOT_SALES'].sum().reset_index()

# Sales by Premium/Non-Premium
premium_sales = merged.groupby("PREMIUM_CUSTOMER")['TOT_SALES'].sum().reset_index()

print("\nSales by Lifestage:\n", lifestage_sales)
print("\nSales by Premium Customer:\n", premium_sales)
```

```
Sales by Lifestage:
                 LIFESTAGE  TOT_SALES
0  MIDAGE SINGLES/COUPLES  184751.30
1            NEW FAMILIES   50433.45
2          OLDER FAMILIES  352467.20
3   OLDER SINGLES/COUPLES  402420.75
4                RETIREES  366470.90
5          YOUNG FAMILIES  316160.10
6   YOUNG SINGLES/COUPLES  260405.30

Sales by Premium Customer:
   PREMIUM_CUSTOMER  TOT_SALES
0           Budget  676211.55
1       Mainstream  750744.50
2          Premium  506152.95
```

Packet Size Analysis

```
# Extract packet size from product name
merged['PACK_SIZE'] = merged['PROD_NAME'].str.extract(r'(\d+)[gG]').astype(float)

# Keep realistic chip sizes
merged = merged[(merged['PACK_SIZE'] >= 90) & (merged['PACK_SIZE'] <= 300)]

# Sales by packet size
packet_sales = merged.groupby('PACK_SIZE')['TOT_SALES'].sum().reset_index()
print(packet_sales.sort_values('TOT_SALES', ascending=False).head())
```

```
   PACK_SIZE  TOT_SALES
9      175.0   485431.4
5      150.0   304288.5
3      134.0   177655.5
1      110.0   162765.4
8      170.0   146673.0
```

```
segment_drivers = merged.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).agg(
    TOTAL_SALES=('TOT_SALES','sum'),
    TXN_COUNT=('TXN_ID','nunique'),
    AVG_SPEND_TXN=('TOT_SALES','mean'),
    AVG_UNITS_TXN=('PROD_QTY','mean')
).reset_index()

print(segment_drivers)
```

```
                 LIFESTAGE PREMIUM_CUSTOMER  TOTAL_SALES  TXN_COUNT  \
0   MIDAGE SINGLES/COUPLES           Budget      31792.8       4645
1   MIDAGE SINGLES/COUPLES       Mainstream      79675.3      10785
2   MIDAGE SINGLES/COUPLES          Premium      52032.6       7555
3             NEW FAMILIES           Budget      19466.4       2758
4             NEW FAMILIES       Mainstream      15130.6       2145
5             NEW FAMILIES          Premium      10168.8       1463
6           OLDER FAMILIES           Budget     149433.9      21228
7           OLDER FAMILIES       Mainstream      92240.7      13101
8           OLDER FAMILIES          Premium      71976.8      10278
9    OLDER SINGLES/COUPLES           Budget     121080.4      16858
10   OLDER SINGLES/COUPLES       Mainstream     118811.3      16847
11   OLDER SINGLES/COUPLES          Premium     116749.7      16233
12                RETIREES           Budget      99795.6      13895
13                RETIREES       Mainstream     138330.0      19742
14                RETIREES          Premium      86215.1      11971
15          YOUNG FAMILIES           Budget     123663.9      17551
16          YOUNG FAMILIES       Mainstream      82861.7      11897
17          YOUNG FAMILIES          Premium      74982.8      10640
18   YOUNG SINGLES/COUPLES           Budget      54946.5       8603
19   YOUNG SINGLES/COUPLES       Mainstream     137527.7      18911
20   YOUNG SINGLES/COUPLES          Premium      37160.5       5816

    AVG_SPEND_TXN  AVG_UNITS_TXN
0        6.804966       1.889983
1        7.346054       1.911580
2        6.843693       1.889517
3        7.022511       1.852814
4        7.040763       1.854816
5        6.936426       1.857435
6        6.977676       1.945461
7        6.983699       1.948592
8        6.932845       1.945290
9        7.148025       1.913100
10       7.006623       1.910479
11       7.154219       1.914210
12       7.144076       1.890543
13       6.975794       1.886283
14       7.172041       1.901090
15       6.995752       1.939469
16       6.914938       1.940666
17       6.992055       1.938083
18       6.367656       1.801136
19       7.249747       1.852240
20       6.365279       1.801131
```

Segment Drivers

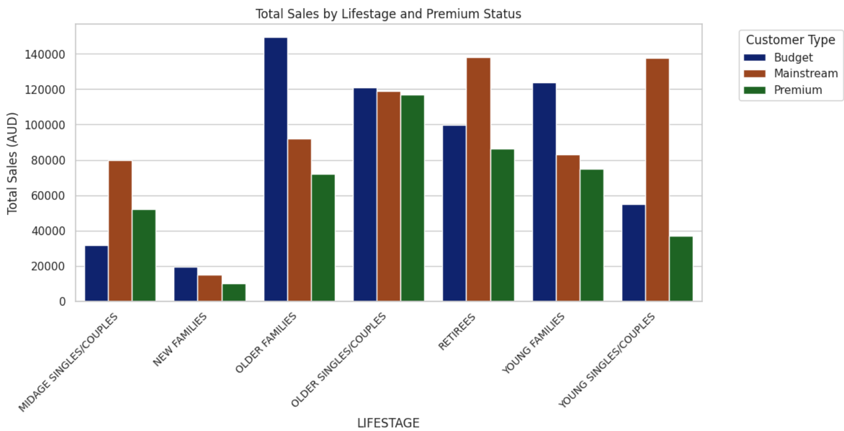
```
# Segment-level drivers of sales
segment_drivers = merged.groupby(['LIFESTAGE','PREMIUM_CUSTOMER']).agg(
    TOTAL_SALES=('TOT_SALES','sum'),
    TXN_COUNT=('TXN_ID','nunique'),
    AVG_SPEND_TXN=('TOT_SALES','mean'),
    AVG_UNITS_TXN=('PROD_QTY','mean')
).reset_index()

# Preview
print(segment_drivers.head())

# Save to CSV
segment_drivers.to_csv("segment_drivers.csv", index=False)
```

```
                LIFESTAGE PREMIUM_CUSTOMER  TOTAL_SALES  TXN_COUNT  \
0  MIDAGE SINGLES/COUPLES           Budget      31792.8       4645
1  MIDAGE SINGLES/COUPLES       Mainstream      79675.3      10785
2  MIDAGE SINGLES/COUPLES          Premium      52032.6       7555
3            NEW FAMILIES           Budget      19466.4       2758
4            NEW FAMILIES       Mainstream      15130.6       2145

   AVG_SPEND_TXN  AVG_UNITS_TXN
0       6.804966       1.889983
1       7.346054       1.911580
2       6.843693       1.889517
```

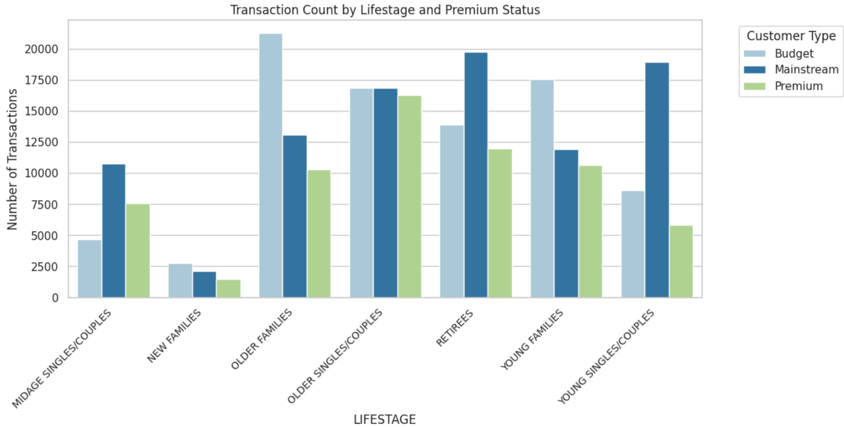
```
3       7.022511       1.852814
4       7.040763       1.854816
```

Visualizations

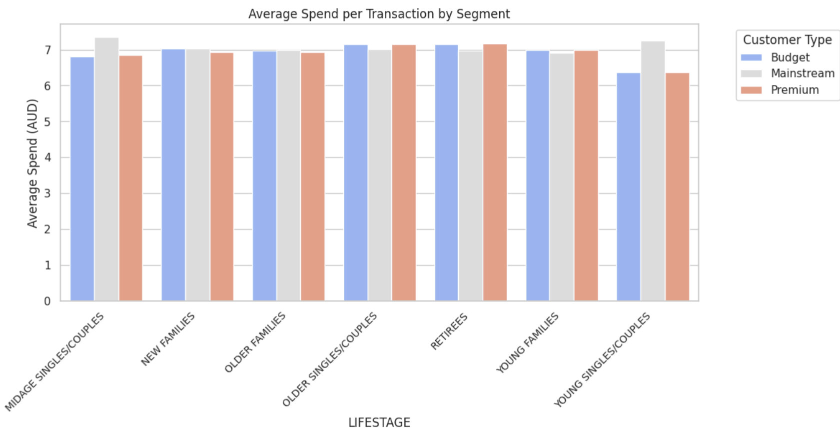
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,6))
sns.barplot(
    x="LIFESTAGE", y="TOTAL_SALES", hue="PREMIUM_CUSTOMER",
    data=segment_drivers, ci=None, palette="dark"
)

# Rotate labels and align them neatly
plt.xticks(rotation=45, ha="right", fontsize=10)

# Move legend outside the plot so it doesn't overlap
plt.legend(title="Customer Type", bbox_to_anchor=(1.05, 1), loc="upper left")

# Add spacing so labels and legend fit in PDF
plt.tight_layout()

plt.title("Total Sales by Lifestage and Premium Status")
plt.ylabel("Total Sales (AUD)")
plt.show()
```




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

plt.figure(figsize=(12,6))
sns.barplot(
    x="LIFESTAGE", y="TXN_COUNT", hue="PREMIUM_CUSTOMER",
    data=segment_drivers, ci=None, palette="Paired"
)

# Rotate labels and align them neatly
plt.xticks(rotation=45, ha="right", fontsize=10)

# Move legend outside the plot so it doesn't overlap
plt.legend(title="Customer Type", bbox_to_anchor=(1.05, 1), loc="upper left")

# Add spacing so labels and legend fit in PDF
plt.tight_layout()

plt.title("Transaction Count by Lifestage and Premium Status")
plt.ylabel("Number of Transactions")
plt.show()
```

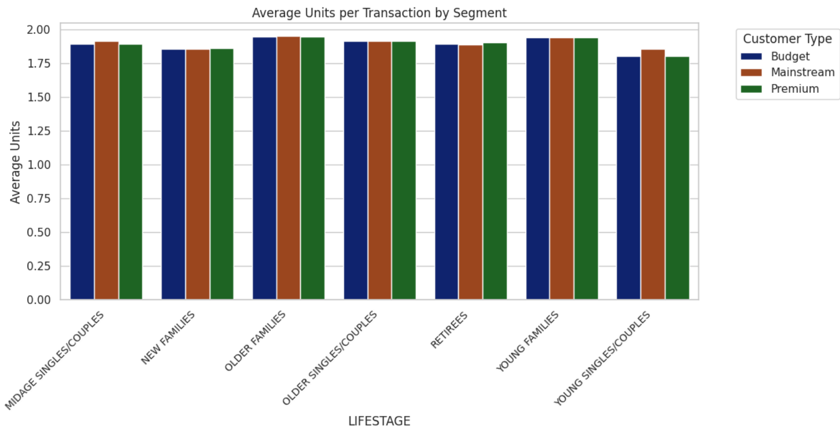



```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,6))
sns.barplot(
    x="LIFESTAGE", y="AVG_SPEND_TXN", hue="PREMIUM_CUSTOMER",
    data=segment_drivers, ci=None, palette="coolwarm"
)

# Rotate labels and align them neatly
plt.xticks(rotation=45, ha="right", fontsize=10)

# Move legend outside the plot so it doesn't overlap
plt.legend(title="Customer Type", bbox_to_anchor=(1.05, 1), loc="upper left")

# Add spacing so labels and legend fit in PDF
plt.tight_layout()

plt.title("Average Spend per Transaction by Segment")
plt.ylabel("Average Spend (AUD)")
plt.show()
```




```
plt.figure(figsize=(12,6))
sns.barplot(
    x="LIFESTAGE", y="AVG_UNITS_TXN", hue="PREMIUM_CUSTOMER",
    data=segment_drivers, ci=None, palette="dark"
```

```
)

# Rotate labels and align them neatly
plt.xticks(rotation=45, ha="right", fontsize=10)

# Adjust legend placement so it doesn’t overlap
plt.legend(title="Customer Type", bbox_to_anchor=(1.05, 1), loc="upper left")

# Add spacing so labels and legend fit in PDF
plt.tight_layout()

plt.title("Average Units per Transaction by Segment")
plt.ylabel("Average Units")
plt.show()
```



Chips Category Analysis – Final Insights Report

1. Data Preparation

- **Transaction Data:**
  - Checked for missing values → none found.
  - Removed 1 duplicate record.
  - Filtered out extreme outliers (e.g., PROD_QTY ≥ 50, TOT_SALES ≥ 100).
  - Verified product categories and extracted packet sizes (90g–300g range).
- **Customer Data:**
  - No missing values or duplicates.
  - Clean dataset ready for merge.
- **Merged Dataset:**
  - Joined on LYLTY_CARD_NBR.
  - Created a clean, analysis-ready dataset linking transactions to customer segments.

2. Core Metrics

- **Total Sales by Lifestage:**
  - Highest contributors: Older Singles/Couples (402k), Retirees (366k), Older Families (352k).
  - Moderate contributors: Young Families (316k), Young Singles/Couples (~260k).
  - Lowest contributor: New Families (50k).
- **Total Sales by Premium Status:**
  - Mainstream (751k) dominates.
  - Budget (676k) is strong.
  - Premium (506k) is lowest.

- Chips are clearly a **mass-market product**.

3. Packet Size Analysis

- **Top Packet Sizes:**
  - 175g (~485k) ⟶ hero product.
  - 150g (~304k) ⟶ strong secondary.
  - Mid-range sizes (134g, 110g, 170g) also contribute significantly.
- **Insight:** Medium-to-large packs (150–175g) are the most popular, aligning with family/group consumption.
- **Smaller packs (110g, 134g)** appeal more to younger singles/couples for snacking.

4. Segment Drivers

- **Older Families, Retirees, Older Singles/Couples** ⟶ consistently high sales and transaction counts.
- **Young Families & Young Singles/Couples** ⟶ moderate sales, lower spend per transaction (~6.3–7 AUD).
- **Premium customers** ⟶ fewer transactions and lower overall sales compared to Budget/Mainstream.
- **Average spend per transaction:** ~7 AUD.
- **Average units per transaction:** ~2 packets.

5. Key Insights

- **Demographics:** Older segments drive the bulk of sales.
- **Customer Type:** Mainstream and Budget dominate; Premium is less relevant.
- **Packet Size:** 175g packs are the most popular, followed by 150g.
- **Behavior:** Customers typically spend ~7 AUD and buy ~2 packets per trip.

6. Recommendations

1. **Target Older Demographics (Retirees, Older Families, Older Singles/Couples):**
   - Promotions on family-size packs (150–175g).
   - Loyalty programs tailored to frequent buyers.
2. **Focus on Mainstream & Budget Customers:**
   - Price-sensitive promotions (multi-buy offers, discounts on 175g packs).
   - Ensure wide availability of popular sizes.
3. **Engage Younger Segments:**
   - Market smaller packs (110g, 134g) as “on-the-go” or “snack-size” options.
   - Position them for impulse purchases.
4. **Anchor Strategy Around 175g Packs:**
   - Hero product for advertising and promotions.
   - Bundle offers with secondary sizes (150g) to maximize basket size.

7. Conclusion

The analysis shows that chips are a **mainstream, family-oriented product**.

- **Older demographics** and **mainstream/budget customers** are the key drivers.
- **175g packs** should be the centerpiece of marketing.
- **Smaller packs** can be leveraged to attract younger singles/couples.